Sommario

337/338 - luglio/agosto 2013

ABBIAMO BISOGNO DEL TUO AIUTO!
Facci conoscere le tue preferenze votando il tuo articolo preferito sul sito **www.farelettronica.com**

Divertiti e metti alla prova le tue conoscenze con **ELETTRO QUIZ** e **vinci** ogni mese **esclusivi premi!**

Editoriale

Domotica fatta in casa

Siamo oramai in pieno clima estivo. Per chi sorseggia una bibita fresca sulla spiaggia non c'è niente di meglio, ma per chi è a casa, il cado e l'umidità prosciugano le nostre energie tanto che alzarsi per spegnere la luce, comporta una fatica enorme… In questo numero estivo vi proponiamo dunque tre soluzioni per il pilotaggio remoto di carichi: una consiste nella gestione di relè tramite bluetooth, una è il Wi-Com presentato il numero scorso, che permette il pilotaggio di relè tramite un tablet Android, la terza è un vero e proprio mini-sistema domotico gestito da Raspberry PI il cui stato viene visualizzato su un normale TV. Che siate al mare o in montagna, siamo sicuri che troverete utili questo tipo di circuiti.

Per i "robottari" presentiamo in anteprima la recensione dell'Arduino Robot: il primo Arduino su ruote ufficiale. Il robot sarà ufficialmente in vendita tra qualche settimana, ma intanto possiamo scoprire tutte le caratteristiche e potenzialità.

Lascio a voi il piacere di scoprire gli altri articoli di questo numero, augurandovi delle serene vacanze. Il prossimo numero di Fare Elettronica vi aspetta online a Settembre.
Buona lettura!

Maurizio Del Corso

DIRETTORE RESPONSABILE
Antonio Cirella

DIRETTORE TECNICO
Maurizio Del Corso

Segreteria di redazione
Giorgia Generali

Comitato scientifico
Simone Masoni (Microtest), Francesco Picchi (Microtest), Massimo Rovini (Università degli Studi di Pisa).

Art director
Patrizia Villa

Hanno collaborato in questo numero:
Silvano Breggion,
Daniele Cappa,
Marco Carminati,
Roberto D'Amico (IW0GTF),
Giovanni Di Maria,
Antonio Giannico,
Giuseppe La Rosa,
Walter Lucetti,
Massimiliano Miocchi,
Giorgio Ober,

Direzione e redazione
INWARE Edizioni srl
Via Giotto, 7 - 20032 Cormano (MI)
Tel. 02.66504794
Fax 02.42101817
info@inwaredizioni.it
www.inwaredizioni.it
Redazione: fe@inwaredizioni.it

Pubblicità per l'Italia
Agostino Simone
Tel. +39 347 2230684
media@inwaredizioni.it

Europe and Americas
Elisabetta Rossi
Tel. +39 328 3245956
international@inwaredizioni.it

Asia
Cybermedia Communications Inc.
Tel. +886-(0)2-2691-2785
asia@inwaredizioni.it

Rest of the world
Inware Edizioni srl
+39 02 66504794
info@inwaredizioni.it

Stampa
Prontostampa
Via Redipuglia 150
24045 Fara Gera d'Adda (BG)

Distribuzione
Parrini & C s.p.a.
Via di Santa Cornelia, 9
00060 Formello (RM)

Ufficio abbonamenti
INWARE Edizioni srl
Via Giotto, 7 - 20032 Cormano (MI)
Per informazioni, sottoscrizione o rinnovo dell'abbonamento:
abbonamenti@inwaredizioni.it
Tel. 02.66504794
Fax. 02.42101817
L'ufficio abbonamenti è disponibile telefonicamente dal lunedì al venerdì dalle 14,30 alle 17,30.
Tel. 02.66504794
Fax 02.42101817
Abbonamento per l'Italia:
€ 55,00
Abbonamento per l'estero:
€ 115,00
Gli arretrati potranno essere richiesti, per iscritto, a **€ 9,00**
oltre le spese di spedizione

Autorizzazione alla pubblicazione
Tribunale di Milano n.647
del 17/11/2003

Mensile associato all'USPI
(Unione Stampa Periodica Italiana)

Collaborare con FARE ELETTRONICA
Le richieste di collaborazione vanno indirizzate all'attenzione di Maurizio Del Corso (m.delcorso@inwaredizioni.it) e accompagnate, se possibile, da una breve descrizione delle vostre competenze tecniche e/o editoriali, oltre che da un elenco degli argomenti e/o progetti che desiderate proporre.

Elenco inserzionisti

Elettroshop
Via Giotto, 7 - 20032 Cormano (MI)
Tel. 02 66504755 - www.elettroshop.com

Grifo
Via dell'Artigiano 8/6 -
40016 San Giorgio Di Piano (BO)
Tel. 051-892052 - www.grifo.it

Innovability
Centro Direzionale Milanofiori Strada 1 Palazzo F2 -
20090 Assago (MI)
Tel. 02 48517925 - www.gowireless.it

Micromed
Via Valpadana 126B/2 - 00141 Roma (RM)
Tel. 06/90024006 - www.micromed.it

MikroElektronika
Visegradska, 1A - 11000 Belgrade
Tel. +381 11 3628830 - www.mikroe.com

Millennium Dataware
Corso Repubblica 48 - 15057 Tortona (AL)
Tel. 0131 860254 - www.mdsrl.it

PCB Pool
Bay 98-99 - Shannon Free Zone -
Shannon - County Clare
Tel. 02 64672645 – www.pcb-pool.com

Tecnoimprese
Via Console Flaminio, 19 - 20134 (MI)
Tel. 02 2101111 – www.fortronic.it

News

Il dispositivo di autenticazione DeepCover di Maxim Integrated protegge i sistemi elettronici

Maxim Integrated Products, Inc. ha annunciato la disponibilità dei campioni di DS28E35 DeepCover Secure Authenticator, una soluzione crittografica ad alta sicurezza che consente ad un controllore host di autenticare le proprie periferiche. Il dispositivo DS28E35 integra un engine ECDSA (Elliptic Curve Digital Signature Algorithm) basato su FIPS 186 per implementare una crittografia asimmetrica (a chiave pubblica) per un protocollo di autenticazione sfida-risposta tra il controllore host e le periferiche, i sensori o i moduli ad esso collegati. Funzionante tramite un singolo pin grazie all'interfaccia 1-Wire, il DS28E35 consente di ridurre la complessità delle interconnessioni, semplificare i progetti e contenere i costi. Il dispositivo offre la sicurezza dell'autenticazione basata su una crittografia robusta in una varietà di applicazioni riguardanti, tra l'altro, sensori medici, moduli PLC (programmable logic controller) in campo industriale, dispositivi consumer.

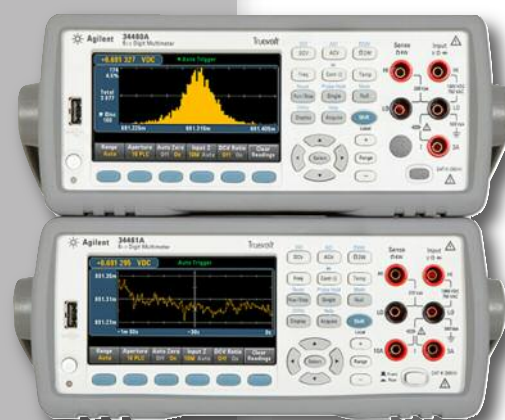
www.maximintegrated.com/datasheet/index.mvp/id/8052

Agilent Technologies presenta i multimetri digitali da 6 1/2 cifre di nuova generazione

Agilent Technologies Inc. presenta i multimetri digitali Serie Truevolt. Si tratta di multimetri digitali che offrono diversi vantaggi rispetto ai modelli precedenti. Aiutano i tecnici a vedere con nuove modalità i risultati delle misure effettuate, permettono di ottenere informazioni utili più rapidamente e di documentare le attività svolte più facilmente. L'esclusiva tecnologia Truevolt riduce l'influenza sui risultati di fattori estranei come il rumore, la corrente iniettata e la corrente di polarizzazione d'ingresso, per migliorare l'intervallo di confidenza delle misure. Il multimetro digitale Agilent 34461A è il diretto sostituto del multimetro digitale di riferimento sul mercato Agilent 34401A ed è stato progettato per semplificare al massimo la migrazione per gli utilizzatori attuali dello strumento 34401A. Il modello Agilent 34460A offre ai tecnici un prodotto di riferimento per cominciare a utilizzare multimetri digitali della classe di strumenti a 6½ cifre.

Rispetto al modello 34401A, il nuovo multimetro digitale 34461A offre una più ampia gamma di correnti da 100 µA a 10 A. Entrambi i modelli 34460A e 34461A offrono funzioni di misura della temperatura (RTD/PT100, termistori) e una funzioni di misura sui diodi potenziate, che permettono di misurare una più ampia tensione di fondo scala (5 V), così da poter collaudare più tipi di diodi, come i LED. La maggior parte dei multimetri digitali mostra i risultati su un display numerico a bassa risoluzione. Con la Serie Truevolt di Agilent, gli utilizzatori possono sfruttare un display ad alta risoluzione a colori da 4,3 pollici che visualizza le misure in formato numerico, sotto forma di tendenze di lungo periodo (il solo modello 34461A), istogrammi e informazioni statistiche sulle misure. E' possibile anche impostare le preferenze del display, che vengono richiamate automaticamente alla successiva accensione dello strumento. Il trasferimento dei file tramite l'interfaccia USB tra un multimetro digitale Agilent Truevolt e un PC non richiede l'installazione di alcun driver e avviene direttamente sfruttando i protocolli di trasferimento standard USB. Gli utilizzatori possono spostare e trasferire i dati delle misure eseguite, le impostazioni dello strumento e le schermate all'interno delle applicazioni sul PC senza richiedere alcun software supplementare. Il software Digital Multimeter Connectivity Utility permette ai tecnici di controllare, catturare e visualizzare i dati misurati con i multimetri digitali Agilent nei loro laboratori. Con un singolo click potranno trasferire i dati a un PC tramite le interfacce USB, GPIB, LAN o RS-232 (quando si collegano i multimetri digitali Agilent delle generazioni precedenti).

Info:
http://www.home.agilent.com/agilent/campaign.jspx?cc=IT&lc=ita&ckey=2351131&id=2351131&cmpid=1-5188745445

Video:
http://www.agilent.us/home/leaving.shtml?http%3A//www.youtube.com/watch%3Fv%3DuomVgFKUQFY

Dimensioni ridotte e prestazioni migliori per i nuovi fotoaccoppiatori Toshiba

Toshiba Electronics Europe annuncia due nuovi fotoaccoppiatori con uscita a Triac caratterizzati da un maggiore isolamento in termini di tensione e di rumore elettrico e disponibili in piccolissimi contenitori SO6 a doppio stampaggio (double-mold). I dispositivi TLP265J e il TLP266J hanno una tensione di isolamento nominale minima di 3750 Veff e sono progettati per l'impiego in azionamenti a triac, controllori programmabili, moduli di uscita in alternata e relè allo stato solido utilizzati in elettro-

News

domestici e altre apparecchiature per la casa. Il modello TLP265J è costituito da un fototriac non a passaggio per lo zero, mentre il modello TLP266J è costituito da un fototriac a passaggio per lo zero. Entrambi sono accoppiati a un LED a infrarossi all'arseniuro di gallio. In termini di caratteristiche elettriche, supportano fino a 7 mA di corrente di innesco del LED, il che permette loro di controllare l'uscita con una più bassa corrente di ingresso, contribuendo così a risparmiare sui consumi. Entrambi i dispositivi sono compatibili con una tensione di rete di 240 V e sopportano una tensione di picco minima di 600 V e una corrente operativa massima di 70 mA.

www.toshiba-components.com

Microchip presenta una nuova famiglia di EEPROM a ID Univoci

Microchip annuncia la disponibilità di una nuova famiglia di dispositivi a EEPROM seriali, con serial number 32 bit univoco pre-programmato, per clienti che necessitino ID univoci nelle loro applicazioni. Per le applicazioni che necessitino ID più lunghi di 32 bit, l'ID univoco può essere esteso a 48 bit, 64 bit, 96 bit, 128 bit e altre lunghezze aumentando il numero di byte letti dalla memoria. Poiché l'ID da 32 bit è unico in questi dispositivi, anche qualsiasi sequenza di bit più lunga è unica.
Questi dispositivi ad EEPROM seriali da 2 Kb sono disponibili in bus standard, come I2C, SPI e Microwire, ed integrano fino a 1.5 Kb di memoria che possono essere usati come una EEPROM standard. In aggiunta, Microchip ha rilasciato un dispositivo a unique-ID I2C da 256 Kb, che, oltre al serial number da 32 bit, contiene anche i MAC address IEEE EUI-48 e IEEE EUI-64.
Questi ID sono in un'area protetta dalla scrittura della memoria, fornendo agli utenti fino a 224 Kb di EEPROM da utilizzarsi nelle loro applicazioni. Microchip assicura che questo ID 32 bit è univoco in tutta l'intera famiglia di dispositivi EEPROM 24AA02UID, 24AA025UID, 11AA02UID, 25AA02UID e 24AA256UID a ID-univoco. Microchip ha anche aggiunto al suo portfolio MAC-address EUI-48 esistente una famiglia con MAC address EUI-64 pre-programmati.
Questi dispositivi EEPROM da 2 Kb sono disponibili nei bus I2C, SPI e UNI/O, che forniscono facile ed economico accesso ai MAC address, e offrono fino a 1.5 Kb di EEPROM che può essere usata per immagazzinare impostazioni di configurazione, oppure come area scratch-pad per bufferizzare piccole quantità di dati. I dispositivi EEPROM seriali 24AA02E64, 24AA025E64, 11AA02E64 e 25AA02E64 hanno un Extended Unique Identifier (EUI) 64-bit integrato che è necessario per identificare gli indirizzi fisici degli hardware della rete.
Questi MAC address integrati consentono ai progettisti di acquistare indirizzi solo quando necessari, e di eliminare la necessità di serializzazione e programmazione.

http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=7046&mid=&lang=en

Peregrine Semiconductor offre il più ampio portfolio di IC integrati per la sintonizzazione RF sul mercato

Peregrine Semiconductor Corporation ha rilasciato 4 nuovi Digitally Tunable Capacitor (DTC) DuNE, creando il più ampio portfolio del mercato di IC integrati per la sintonizzazione RF. I DTC PE64906, PE64907, PE64908, e PE64909 contano su una ampio range capacitivo di 0.6 – 7.7 pF e sopportano il trattamento di potenze fino a 34 dBm su un carico di 50 Ohms (30 Vpk RF). I DTC monolitici includono generazione bias integrata, filtro RF e bypass, interfaccia di controllo, e protezione ESD di 2 kV HBM, il tutto in un package di 2 x 2 mm. I dispositivi offrono una soluzione di facile utilizzo per sintonizzazione di antenne e accordatura di impedenze, cosi come filtri accordabili e variatori di fase nei mercati della Radio Frequency Identification (RFID), radio militari, Test-and-Measurement (T&M), Machine-to-Machine (M2M) e infrastrutture wireless.

www.psemi.com/index.php

Alimentatori ad alta efficienza ultra-compatti

ROHM ha presentato degli alimentatori compatti ad alta efficienza, adatti per memorie DDR, microcontrollori e altri componenti destinati alle applicazioni automotive. La serie

News

BD905xx integra un circuito di compensazione di fase e una resistenza di feedback e, rispetto ai circuiti di alimentazione tradizionali, richiede un numero decisamente inferiore di parti esterne; ne consegue una semplificazione della progettazione e una maggiore miniaturizzazione del prodotto finale. La crescente diffusione dei veicoli elettrici e ibridi registrata in questi ultimi anni ha comportato la creazione di sistemi elettronici più sofisticati e un aumento della domanda di microcontrollori e memorie. Normalmente si utilizzano regolatori LDO tradizionali che, però, sono poco efficienti e non sono in grado di soddisfare l'esigenza di correnti più elevate. Per questo motivo si tende a preferire l'utilizzo di convertitori DC/DC, che garantiscono un rendimento maggiore e riescono a gestire correnti più alte. Ma anche questi dispositivi presentano alcuni svantaggi: ad esempio, hanno bisogno di un numero maggiore di componenti esterni e, di conseguenza, occupano più spazio e rendono la progettazione del circuito più complessa. ROHM, invece, è riuscita a ottimizzare il circuito di compensazione all'interno del componente, riducendo dell'80% il numero dei componenti esterni e contribuendo a creare sistemi automotive più compatti.

www.rohm.com/web/eu/home

GMB HR246: Grifo Mini BLOCK - Housing, 24 Opto-In, 16 Relè

La GMB HR246 è un modulo da Barra DIN in grado di alloggiare una CPU Mini Modulo del tipo CAN o GMM da 28 pin o da 40 pin. In questa famiglia sono comprese varie famiglie di CPU come quelle Core 8051, AVR, PIC, ARM7, ecc.La GMB HR246 è completa di 24 ingressi, galvanicamente isolati e visualizzati tramite LED, 16 uscite a Relè visualizzati tramite LED, uscita RTC visualizzata tramite LED, connettore AMPMODU II, da 4+4 vie, per comunicazioni seriali, alimentatore stabilizzato, linee di A/D, PWM, D/A, ecc. In un ingombro minimo, contiene un completo controllore in grado di gestire completamente una specifica applicazione.

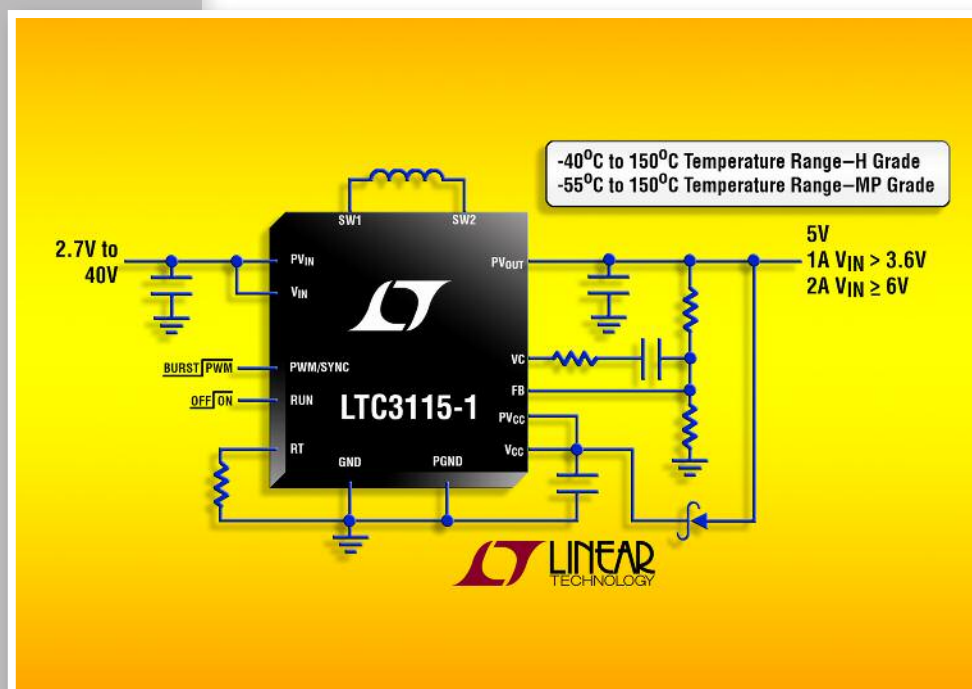
La GMB HR246 è distribuita da Grifo - Via dell'Artigiano 8/6 - 40016 San Giorgio di Piano (BO)

www.grifo.it

News

Convertitore DC/DC buck-boost sincrono da 40V, 2A disponibile nelle versioni per alte temperature (grado H) e ad elevata affidabilità (grado MP)

Linear Technology annuncia la versione per alte temperature, grado H (temperatura di giunzione da -40°C a 150°C) e ad elevata affidabilità, grado MP (da -55°C a 150°C) dell'LTC3115-1 in un package TSSOP da 20 pin con funzionalità termiche avanzate. L'LTC3115-1 è un convertitore buck-boost sincrono che fornisce fino a 2A di corrente continua in uscita da più fonti di alimentazione, dalle batterie agli ioni di litio a singola cella ai rail industriali da 24V/28V, fino agli ingressi da 40V delle applicazioni automotive. L'intervallo di tensione di ingresso e uscita del dispositivo, compreso tra 2,7V a 40V, assicura un'uscita regolata con ingressi superiori, inferiori o uguali all'uscita regolata. La topologia buck-boost a basso rumore incorporata nell'LTC3115-1 assicura una transizione jitter-free continua tra le modalità buck e boost, caratteristica che lo rende un dispositivo particolarmente adatto per i sistemi a radiofrequenza e per altre applicazioni sensibili al rumore che devono mantenere una tensione in uscita costante a basso rumore con un'alimentazione in ingresso variabile. L'efficienza fino al 95% prolunga la durata della batteria nei sistemi con questo tipo di alimentazione. La frequenza di commutazione dell'LTC3115-1 può essere programmata dall'utente tra 100kHz e 2MHz e può essere sincronizzata su un clock esterno. Il circuito PWM proprietario buck-boost assicura un basso rumore e un'elevata efficienza, il tutto riducendo al minimo le dimensioni dei componenti esterni.

www.linear.com/product/LTC3115-1

Eventi

zeroEmission

ZERO EMISSION ROME 2013

Giunto alla sua sesta edizione, ZEROEMISSION ROME è l'evento di riferimento per tutte le aziende e gli operatori interessati allo sviluppo delle energie rinnovabili, all'emission trading e alla sostenibilità ambientale in Italia e nel grande e promettente mercato del bacino del Mediterraneo. ZEROEMISSION ROME 2013 è l'insieme di eventi specializzati dedicati all'energia eolica, all'energia fotovoltaica, al solare termodinamico, all'emission trading, cambiamenti climatici e CCS, agroenergie e biocarburanti. Insieme occuperanno ben quattro grandi padiglioni di Fiera di Roma su un'area di oltre 40.000 metri quadri.

Dove: Roma • Quando: 9-11 Ottobre 2013 • Orari: dalle 9.30 alle 18.30 • Organizzazione: ZeroEmission • info: www.zeroemissionrome.eu

SAVE
Mostra Convegno delle Soluzioni e Applicazioni Verticali di Automazione, Strumentazione, Sensori.

AUTOMAZIONE IN FIERA

SAVE è un appuntamento innovativo che unisce una parte espositiva in fiera ad una forte componente formativa. Area espositiva dove incontrare agli stand i principali leader di settore, centinaia di convegni e workshop accessibili per gli operatori qualificati, cinque eventi internazionali e un evento speciale in contemporanea, gli operatori professionali accedono gratuitamente alla mostra e usufruiscono di tutti i servizi. L'esigenza percepita oggi è rendere adeguate le modalità fieristiche, renderle attuali e sempre più proficue per creare una fiera sull'automazione e sulla strumentazione utile agli operatori.

Dove: Verona • Quando: 29-30 ottobre 2013 • Orari: dalle 9.00 alle 18.00 • Organizzazione: EIOM • info: www.exposave.com

MOSTRA NAZIONALE MERCATO RADIANTISTICO

Mostra mercato radiantistico dell'elettronica, CD, editoria specializzata, telefonia cellulare.

Dove: Montichiari (BS) • Quando: 31 Agosto – 1 settembre 2013 • Orari: dalle 9.00 alle 18.00 • Organizzazione: Centrofiera • info: www.centrofiera.it

ELECTRONIC DAYS

Fiera dell'Elettronica in genere, Mostra Mercato con vendita consentita.

Dove: Montesilvano (PE), presso Palacongressi • Quando: 21-22 Settembre 2013 • Orari: dalle 9 alle 19 • Organizzazione: ITALFIERE srl •info: www.italfiere.net

Eventi

RADIANT AND SILICON

Sin dalle primissime edizioni ha goduto del patrocinio della sezione A.R.I. (Associazione Radioamatori Italiani) di Milano, del supporto dell'A.I.R.E. (Associazione Italiana per la Radio d'Epoca), dell'A.I.R. (Associazione Italiana Radioascolto) e della prestigiosa partecipazione di importanti Enti Pubblici come la Provincia di Milano, presente con le sue postazioni mobili del Servizio di Protezione Civile ed, ultimamente, il Ministero delle Comunicazioni.

La Rassegna si configura in tre settori: mostra-mercato (apparati e componenti per telecomunicazioni, Internet e ricetrasmissioni di terra e satellitari, antenne, elettronica, informatica, telefonia statica e cellulare, hobbistica ed editoria); borsa-scambio (surplus radioamatoriale, informatico ed elettronico); radioantiquariato (radio d'epoca, grammofoni, valvole, ricambi, dischi in vinile, schemari ed editoria specializzata).

Attualmente RADIANT, che si sviluppa su una superficie espositiva superiore ai 10.000 mq. coperti, con più di 180 espositori per edizione ed oltre 40.000 visitatori annui, è considerata la più importante Mostra-Mercato nazionale del suo settore. Fra le categorie merceologiche ricordiamo: elettronica, informatica, telefonia, radiantismo, editoria, tv satellitare, hobbistica, surplus, radio d'epoca.

Dove: Parco Esposizioni Novegro, Segrate (MI) • Quando: 21-22 Settembre 2013 • Orari: Sabato 9.00/18.00 - Domenica 9.00/17.00 • Organizzazione: COMIS • info: www.parcoesposizioninovegro.it

DESIGNING WITH FREESCALE 2013

La serie dei seminari Designing With Freescale offre sessioni di training sulle ultime tecnologie e soluzioni Freescale pre dare agli utenti tutti gli strumenti e le informazioni necessarie per progettare e sviluppare applicazioni con Freescale.

I seminari sono tenuti dai FAE Freescale che rimangono a disposizione dei partecipanti per rispondere ai loro quesiti.

Dove: Milano • Quando: 2 ottobre 2013 • info: www.freescale.com

ITN 2013

A ITN 2013 forte attenzione sarà data allo stato dell'arte e alle attuali opportunità per il comparto attraverso workshop dedicati, in cui verranno presentati numerosi esempi di best practice, dove la parola d'ordine sarà "intelligenza": nel progettare, nell'ottenere e gestire le informazioni, nell'utilizzare le infrastrutture digitali utili alla creazione delle future smart city.

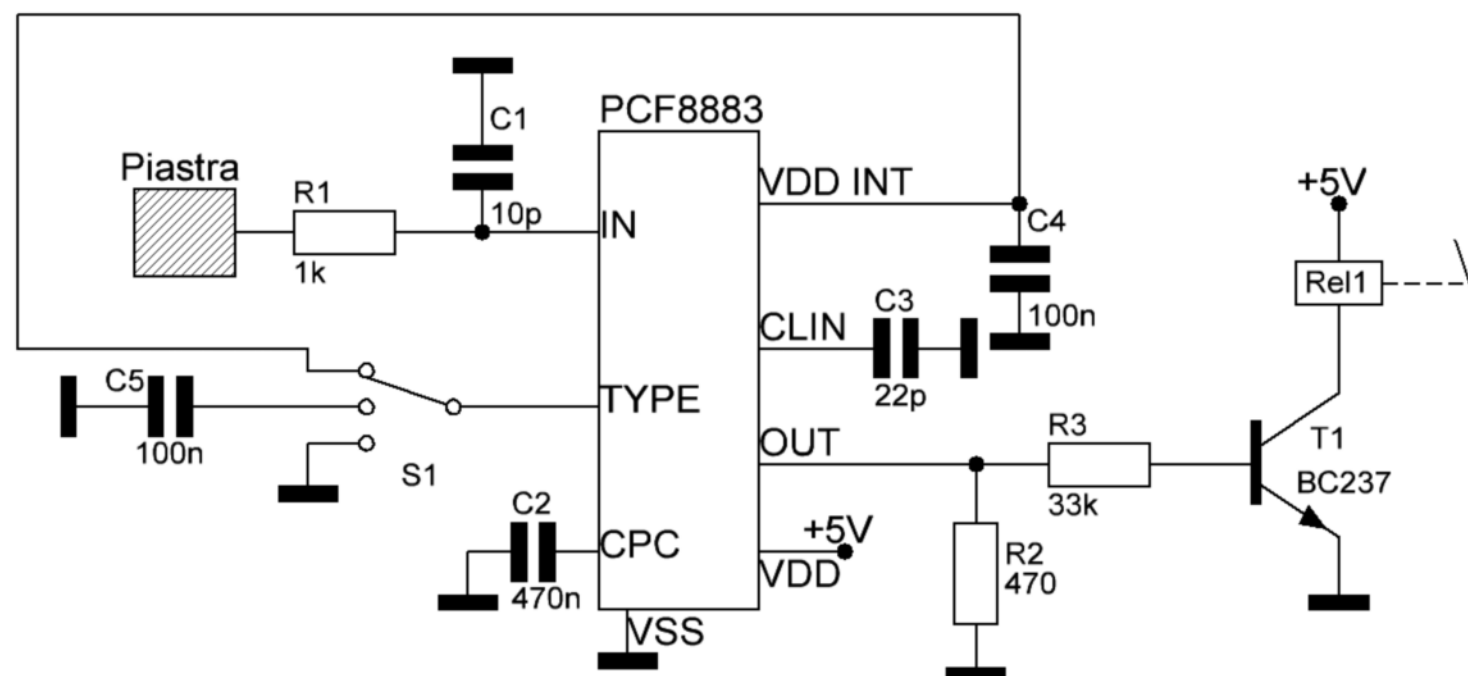
Temi caldi della nuova edizione saranno piattaforme Cloud e soluzioni Big Data come componenti tecnologiche fondamentali per lo sviluppo dei servizi digitali per Smart City.

A ITN si scoprirà come Cloud e Big Data sono rispettimanente il "motore" che spinge e il "cruscotto" che controlla del nostro veicolo-città.

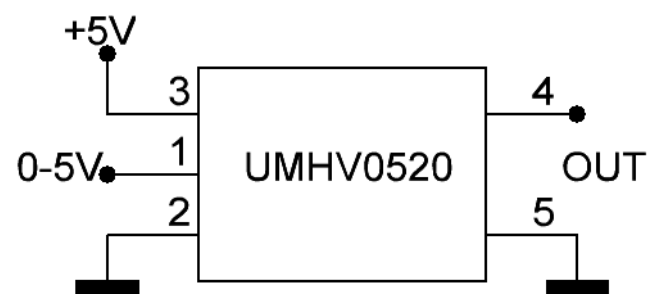
Dove: Torino - Lingotto Fiere • Quando: 26-27 settembre 2013 • info: www.itnexpo.it

Idee di progetto

MIXER SIMMETRICO PER APPARECCHIATURE RADIO

Il circuito rappresentato in figura è un mixer simmetrico utilizzato per frequenze non superiori ai 200MHz. Il circuito deve essere alimentato con una tensione non superiore ai 15V. Per il suo funzionamento non richiede particolari elementi passivi. Tale circuito può essere impiegato per applicazioni radiofoniche.

INTERRUTTORE SENSIBILE ALLE DITA

Il circuito rappresentato in figura è un sensore di prossimità capacitivo impiegato per applicazioni ad uso domestico o industriale. Tale integrato ha tre diverse modalità di funzionamento: monostabile, bistabile e stabile. All'uscita dell'integrato è connesso un transistor NPN in grado di pilotare un relè il quale a sua volta può pilotare carichi più alti.

Idee di progetto

DC-DC A 2000V

Nella figura seguente è rappresentato un semplice circuito realizzato con un integrato dc-dc denominato UMHV0520. Questo integrato deve essere alimentato con una tensione non superiore ai 5V ed è in grado di fornire in uscita una tensione variabile da 0 ai 2000V . Tale regolazione viene fatta mediante il pin di Program, al quale viene mandata una tensione variabile da 0 e 5V

CONVERTITORE A/D A 4 USCITE

Il circuito proposto è un convertitore D/A realizzato con un MCP4728. Il vantaggio di tale circuito è quello di avere quattro uscite pilotate internamente da un'interfaccia I2C, che può essere connessa direttamente a un microcontrollore oppure collegata ad ARDUINO.

Idee di progetto

COMPANDER

Il circuito seguente è un riduttore di rumore denominato Compander, impiegato spesso in sistemi audio in cui vi è la presenza di forte rumore sonoro. Tale sistema amplifica i segnali deboli , portandoli ad un volare prestabilito e in fare di riproduzione gli riabbassa. Questi sistemi erano spesso utilizzati in impianti audiocassetta per mascherare il fruscio del nastro magnetico.

STABILIZZATORE A 2,5V CON CURRENT SENSE

Lo schema in figura è uno stabilizzatore realizzato con un LM4132, che è in grado di erogare 2,5V con basso Dropout, in più tale circuito offre la funzionalità di avere un sistema di current sense realizzato con dei partitori. Il trimmer posto tra i due resistori offre la possibilità di regolare la sensibilità.

di NICOLA TARASCHI

LA PNEUMATICA CON IL CUBLOC

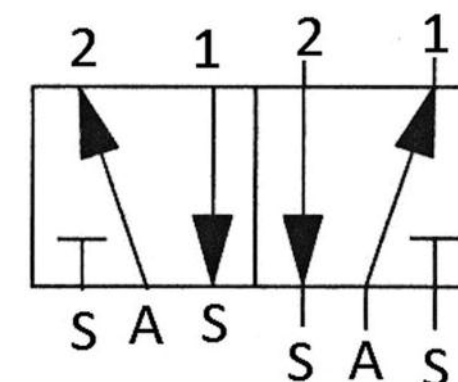
Un esempio di applicazione alla pneumatica dei microcontrollori con l'impiego del PC per l'elaborazione e la memorizzazione dei dati

figura 1: il pistone pneumatico

Nel campo dell'automazione la pneumatica occupa una posizione significativa per la vastità delle applicazioni industriali. L'aria compressa fornisce l'energia per svolgere processi di fabbricazione, di assemblaggio e di movimentazione. L'impiego dei microcontrollori comporta anche in questo campo considerevoli vantaggi nel controllo dei cicli pneumatici . In questo ambito i moduli CUSB della COMFILE, disponendo di connessioni dirette ai componenti elettropneumatici, trovano una collocazione, si può dire ideale.

I COMPONENTI PNEUMATICI

Il componente fondamentale della pneumatica è il pistone (figura 1), nella versione a doppio effetto, il cui movimento lineare è ottenuto con l'afflusso da un lato di aria compressa e il deflusso sull'altro lato e viceversa. Le valvole distributrici hanno il compito di alimentare opportunamente i pistoni, come la valvola 5/2, (lo schema nella figura 2) i cui significato: è 5 vie 2 posizioni. A seconda di quale delle due posi-

figura 2: la valvola 5/2

zioni assume la valvola variano le vie coinvolte.

Nel primo caso l'aria giunge al pistone dall'alimentazione A e va alla via 1 che alimenta il pistone nel movimento da destra a sinistra, mentre l'aria presente nella parte opposta del cilindro va allo scarico attraverso la via 2-S. Nel secondo caso avviene l'opposto: l'aria affluisce da A a 2 e si scarica da 1 a S. Il comando delle posizioni della valvola, avviene con 2 elettrovalvole, generalmente a 24 V DC (figura 3).

La posizione finale del pistone nel movimento lineare è determinata dai finecorsa, che segnalano il raggiungimento della posizione estrema del pistone. La posizione

figura 3: la valvola di distribuzione 5/2 con le due elettrovalvole

di finecorsa viene acquisita da finecorsa meccanici, come nella figura 4: il contatto elettrico viene aperto o chiuso dalla forza della testa del pistone che fa leva sulla rotella. Attualmente il finecorsa meccanico viene più diffusamente sostituito dal tipo magnetico con relè reed: in questo caso il pistone magnetico apre o chiude un contatto reed posto all'esterno del pistone. Il circuito pneumatico si completa con pulsanti che determinano l'avvio o lo stop del processo, e con elementi logici che effettuano operazioni booleane. Ad esempio il movimento di un pistone potrebbe essere avviato solo se due pulsanti vengono schiacciati contemporaneamente, con una operazione di AND, oppure se uno solo dei due pulsanti viene schiacciati(operazione di OR). Quando il circuito pneumatico diventa complesso deve essere compiuto uno studio attento per evitare interferenza dei segnali provenienti dai finecorsa (segnali bloccanti). Lo studio dei segnali bloccanti, come dei segnali ripetuti comporta una maggiore complessità del sistema. Non possiamo approfondire questo argomento ma ci limitiamo a dire che il sistema realizzato con soli componenti pneumatici o elettropneumatici presenta il limite di tutti i sistemi di automazione tradizionali rispetto a quelli realizzati con microcontrollore. Il collegamento diretto fra gli ingressi e le uscite comporta infatti anche l'implementazione di tutte le funzioni logiche necessarie. La realizzazione fisica della funzione OR fra due segnali d'ingresso necessita, ad esempio, l'impiego di una valvola apposita. La temporizzazione dei segnali comporta l'impiego di valvole temporizzatrici, facilmente sostituite dalle funzioni software come quelle di DELAY. Il microcontrollore elimina tutta la necessità di implementare componenti logici, sostituendo il software all'hardware e quindi semplifica lo studio dell'impianto. Lo stesso eventuale pulsante di START o di STOP pneumatico può essere facilmente sostituito da un più comune ed economico pulsante elettrico. Tutti i segnali di input arrivano infatti al microcontrollore e la loro gestione, con la realizzazione dei segnali di uscita opportuni è affidata al software. Nel campo pneumatico i segnali sono generalmente di tipo digitale ON-OFF e il linguaggio LADDER che gestisce situazioni di tipo booleano può risolvere processi di complessità non elevata. Il LADDER è un linguaggio derivato dalla implementazione degli schemi elettrici di automazione e realizza funzioni logiche booleane fra ingressi ed uscite come anche funzioni di conteggio e di temporizzazione e, in versioni più evolute, funzioni sulle grandezze analogiche. Il LADDER assomiglia ad un linguaggio macchina, tipico delle prime applicazioni dei PC, e comporta una stesura del programma poco leggibile e non intuitive, e comunque poco potente. I microcontrollori odierni, che hanno risorse hardware sempre più potenti, in termini di porte digitali ed analogiche di input-output,di memoria RAM ed EEPROM, e soprattutto la disponibilità di potenti linguaggi di programmazione come il BASIC o il C permettono una gestione più efficiente e più "umana" della logica di controllo. Inoltre la connessione seriale con il PC permette una interazione fra le risorse del microcontrollore e quelle del PC, che mette a disposizione l'interfaccia utente e le capacità di memorizzazione e trasmissione dei dati . Attraverso il PC possiamo memorizzare e richiamare diverse soluzioni di logica pneumatica in modo molto più interattivo che con il solo microcontrollore.

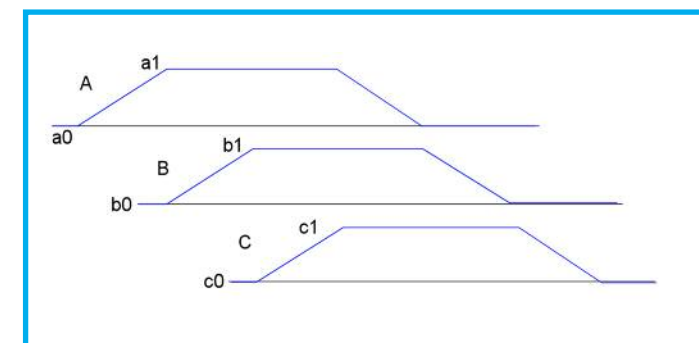



figura 4: il finecorsa

figura 5: la sequenza

Il microcontrollore scelto in questo è il CUBLOC o meglio i modelli CUSB 14R o 22R e simili. Il modulo CUSB si presenta come un controllore ideale per i circuiti pneumatici. Presenta infatti un numero più o meno elevato, a seconda dei modelli, di porte di ingresso e uscita a 24 VDC, tipiche delle applicazioni di elettropneumatica. In questo caso la connessione dei fine corsa o delle elettrovalvole si limita al collegamento dei soli fili, senza altri componenti. Nondimeno la presenza di porte analogiche rende comunque la disponibilità di gestione di segnali analogici presenti nel processo di automazione. L'alimentazione diretta dalla rete a 220 elimina la necessità di un alimentatore. Con una spesa di poco più di 100 euro il CUSB può gestire direttamente 14 fra ingressi ed uscite e quindi già molte applicazioni. E per i vecchi no-

stalgici del LADDER il CUBLOC è uno dei pochi microcontrollori che rende disponibile la programmazione anche in questo linguaggio. Facendo un bilancio economico il maggior peso economico del controllore viene ampiamente annullato dal risparmio in termini di componenti pneumatici , del tempo per lo studio del ciclo, e dei vantaggi nel passare da un tipo di ciclo ad un altro. La ricollocazione fisica e logica dei componenti pneumatici, quando il ciclo iniziale subisca dei cambiamenti, può essere brillantemente sostituito spesso solo da modifiche nel software.

IL CICLO PNEUMATICO

Il ciclo pneumatico si basa sul movimento dei pistoni in base ai segnali proveniente dai fine corsa o dai pulsanti dell'operatore. Il movimento del pistone è avanti o indietro e viene contrassegnato con una lettera che indica il pistone ed il segno + =avanti, - =indietro. Una sequenza possibile, dove ogni singolo passo verrà denominata FASE è la seguente:

A+	A-
B+	B-
C+	C-

La sequenza è regolata dal consenso dei finecorsa. Supponiamo che a1 e a2 siano rispettivamente il fine corsa iniziale e finale del pistone A, b1,b2 quelli relativi al pistone B, c0,c1 del pistone C. L'inizio del ciclo avviene se è anche presente un segnale di START, proveniente da un pulsante. Il ciclo si ripete fintantoché è presente il segnale di START. La sequenza che si realizza nelle 6 fasi deve pertanto rispettare le seguenti condizioni logiche:

1. il pistone A va avanti se il finecorsa c0 è ON e se il pulsante di START è ON, si ferma quando a1 è ON
2. Il pistone B si muove in avanti se il fine corsa a1 è ON e si ferma quando b1 è ON
3. Il pistone C si muove se il fine corsa b1 è ON e si ferma quando c1 è ON
4. il pistone A torna indietro se il finecorsa c1 è ON e si ferma se a0 è ON
5. Il pistone B torna indietro se il fine corsa a0 è ON e si ferma quando b0 è ON
6. Il pistone C torna indietro se il finecorsa a0 è ON e si ferma quando c0 è ON

Il ciclo continua proseguendo dall'inizio. In definitiva il movimento di ogni pistone inizia quando il finecorsa che segna la fine della fase precedente è ON e termina quando è ON il finecorsa della sua corsa. La condizione per l'avvio della fase successiva è quindi il completamento della fase precedente. La figura 5 rappresenta schematicamente la successione del movimento dei pistoni, dove la linea inclinata verso l'alto indica la corsa +, pistone che esce fuori, mentre la linea inclinata verso il basso rappresenta la corsa -,il pistone che rientra.

IL SOFTWARE SUL CUBLOC

Il software sul CUBLOC consiste in 2 programmi. Il primo accetta attraverso la comunicazione seriale tutti i dati relativi alla sequenza pneumatica dal PC e li memorizza nella sua EEPROM (listato 1).
Il secondo programma è quello che realizza il ciclo.
Vengono innanzi tutto definiti i vettori VFC1,VFC2,VELV che rappresentano le porte associate a:

LISTATO 1
Il programma di comunicazione

Download

```
. .
Set Debug Off
psz=0
Opencom 0,9600,18,100,100
Bclr 0,2
On Recv0 Gosub ricevo
Count1=0
lungh=40
Do
If Count1= lungh Then
esamina
Count1=0
End If
Loop
ricevo:
If Blen(0,0)>0 Then
aa=Get(0,1)
Count1=Count1+1
vett(Count1)=aa
End If
Return
Sub esamina()
If vett(1)=71 Then
kk=0
Delay 250
For j=2 To 4
For i=1 To 10
kk=j+4*(i-1)
aa=vett(kk)-64
scrivi_ee
Next
Next
End If
End Sub
Sub scrivi_ee()
psz=psz+2
Eewrite psz,aa,2
End Sub
#include "libreria. cub"
```

- il finecorsa che attiva l'elettrovalvola
- il finecorsa che ferma l'elettrovalvola
- L'elettrovalvola

Ed inoltre il pulsante di stop (variabile PULSANTE_STOP).
La variabile NUMFASI memorizza il numero di fasi. Successivamente attraverso la subroutine LEGGI_EE legge i dati dalla EEPROM, precedentemente trasmessi dal PC, e li memorizza nei vettori prima definiti.

Successivamente esegue il ciclo con i dati ricevuti dal PC. Il programma vero e proprio (listato 2) si impernia sulla subroutine FAI_FASE che compie le azioni legate ad ogni fase.

Sub fai_fase(FC1 As Integer,FC2 As Integer,PIST As Integer)
passando i parametri:

- il primo parametro FC1 è il fine corsa che attiva l'elettrovalvola

- il secondo parametro FC2 è il fine corsa che ferma l'elettrovalvola
- il terzo parametro PIST è la porta associata alla elettrovalvola

La subroutine legge lo stato della porta associata al primo fine corsa e se è ON attiva con il comando OUT PIST,1 la porta associata alla elettrovalvola.
Se è ON il secondo fine corsa arresta l'elettrovalvola con il comando OUT PIST,0 e contemporaneamente aumenta la variabile FASE che quindi consente lo svolgimento della fase successiva (listato 2).

IL SOFTWARE CON IL PC

L'utilizzo del PC comporta possibilità di controllo ed una interfaccia utente ben maggiori. In questo caso il software permette di definire le porte associate al ciclo e la sequenza, quindi determina la sequenza dei segnali di INPUT ed OUTPUT e invia i dati al CUBLOC. la definizione del ciclo avviene con un comune file di testo che inizia con la definizione delle porte di uscita associate alle elettrovalvole che comandano le corse avanti e indietro di ogni pistone.

Prima riga=41,42,43,44,45,46

Quindi al pistone A sono associate le elettrovalvole 41(corsa avanti) e 42(corsa indietro) e analogamente per i pistoni B e C. La seconda riga riporta le porte di ingresso associate ai fine corsa. Quindi l'ingresso 11 è associato al finecorsa a1 del primo pistone, l'ingresso 12 al finecorsa a2 dello stesso pistone e così via.

Seconda riga=11,12,13,37,38,39

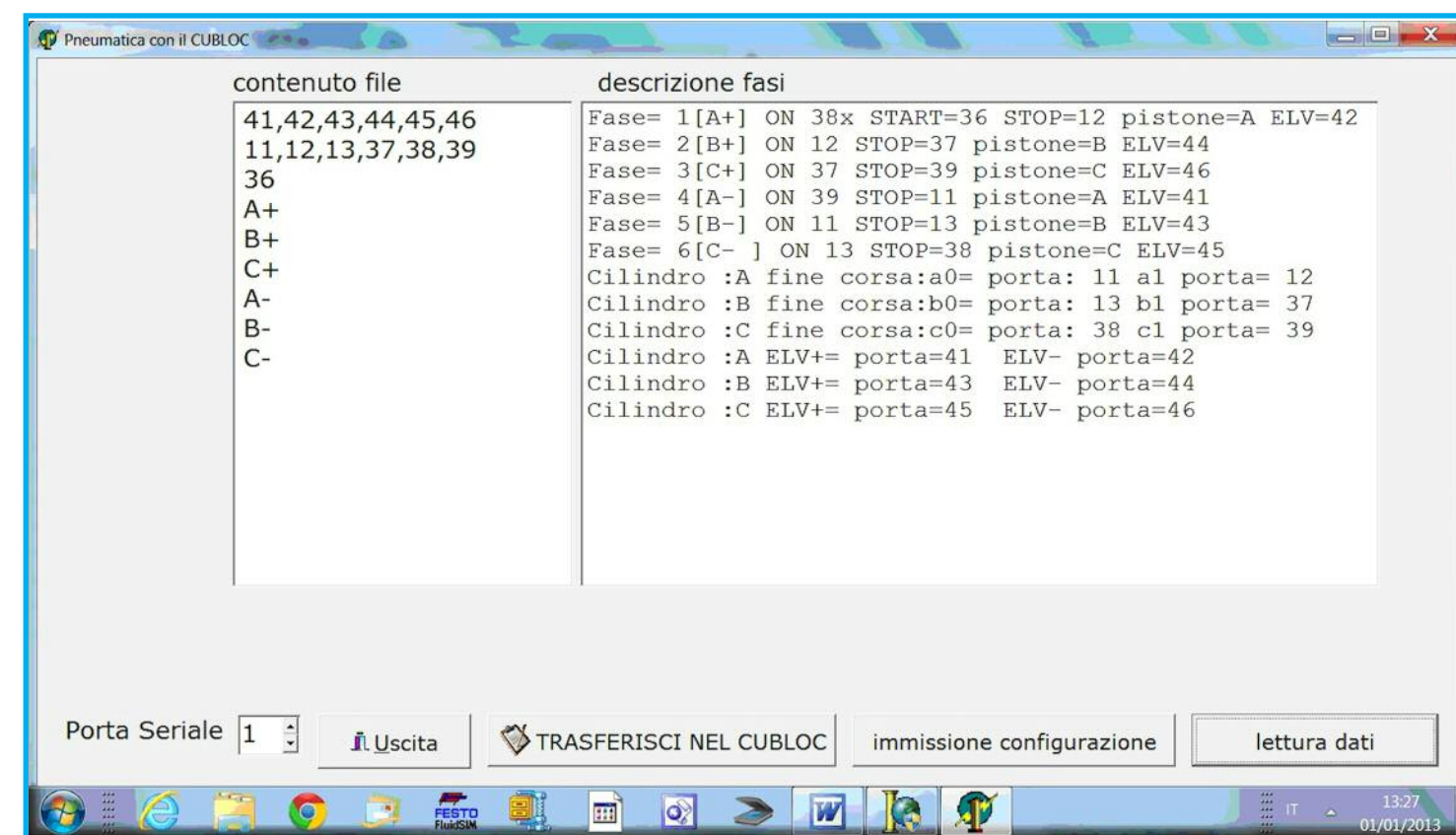
LISTATO 2

Il programma di esecuzione del ciclo

```
Const Device = CB280
Dim pulsante_stop     As  Inte-
ger
Dim fase As Integer
Dim numfasi As Integer
Dim vfc1(10) As Integer
Dim vfc2(10) As Integer
Dim velv(10) As Integer
. .
Set Debug On
Debug clr
leggi_ee
fase=1
Do
Debug clr
stop=In(pulsante_stop)
If stop=1 Then resetta
fai_fase vfc1(fase-1),vfc2(fa-
se-1),velv(fase-1)
Delay 100
Loop
Sub fai_fase(fc1 As Integer,fc2
As Integer,pist As Integer)
a=In(fc1)
If a=1 Then
Out pist,1
SP=Dec(pist)
SP="pistone av/ind="+SP
scrivist 9,SP
End If
a=In(fc2)
If a=1 Then
Out pist,0
SP=Dec(pist)
SP="pistone stop="+SP
scrivist 10,SP
If fase<numfasi Then
fase=fase+1
Else
fase=1
End If
End If
End Sub
Sub leggi_ee()
psz=0
For i=1 To 6
psz=psz+2
aa=Eeread(psz,2)
aa=cercainp(aa)
b=3+(i-1)*3
psz=psz+2
aa=Eeread(psz,2)
aa=cercainp(aa)
b=4+(i-1)*3
psz=psz+2
aa=Eeread(psz,2)
aa=cercaout(aa)
b=5+(i-1)*3
Next
End Sub
```

Download

```
num fasi=          6.
stop=         36.
fc1=         38.
fc2=         12.
elv=         42.
fc1=         12.
fc2=         37.
elv=         44.
fc1=         37.
fc2=         39.
elv=         46.
fc1=         39.
fc2=         11.
elv=         41.
fc1=         11.
fc2=         13.
elv=         43.
fc1=         13.
fc2=         38.
elv=         45.
```

figura 6: i dati ricevuti dal cubloc

```
stop=           0.
fase=           3.
fine corsa AV      =      37.
fine corsa STOP    =      39.
ELV                =      46.

pistone av/ind=46
```

figura 7: lo svolgimento del programma sul cubloc

Pneumatica con il CUBLOC

contenuto file

```
41,42,43,44,45,46
11,12,13,37,38,39
36
A+
B+
C+
A-
B-
C-
```

descrizione fasi

```
Fase= 1[A+] ON 38x START=36 STOP=12 pistone=A ELV=42
Fase= 2[B+] ON 12 STOP=37 pistone=B ELV=44
Fase= 3[C+] ON 37 STOP=39 pistone=C ELV=46
Fase= 4[A-] ON 39 STOP=11 pistone=A ELV=41
Fase= 5[B-] ON 11 STOP=13 pistone=B ELV=43
Fase= 6[C- ] ON 13 STOP=38 pistone=C ELV=45
Cilindro :A fine corsa:a0= porta: 11 a1 porta= 12
Cilindro :B fine corsa:b0= porta: 13 b1 porta= 37
Cilindro :C fine corsa:c0= porta: 38 c1 porta= 39
Cilindro :A ELV+= porta=41  ELV- porta=42
Cilindro :B ELV+= porta=43  ELV- porta=44
Cilindro :C ELV+= porta=45  ELV- porta=46
```

Porta Seriale 1 | Uscita | TRASFERISCI NEL CUBLOC | immissione configurazione | lettura dati

figura 8: il menu del programma

figura 9: il gruppo pneumatico complessivo

Terza riga: il pulsante START è associato alla porta di ingresso 36

Definite le porte segue l'elenco delle fasi che compongono il ciclo. La definizione delle fasi avviene con i comandi tipici della simbologia pneumatica A+= pistone A avanti, oppure A -=pistone A indietro e così per gli altri pistoni. Il PC elabora i dati e costruisce la sequenza delle fasi determinando i segnali che comandano la fase. Al contenuto simbolico del file viene quindi associata la descrizione delle fasi operative riportando per ciascuna i fine corsa e le relative porte di INPUT, i pistoni, le porte d'uscita (figura 8).

Nella figura 8 viene riportata la videata del programma sul PC. Il pulsante: lettura dati, consente la lettura del file di testo, riportata nell'area denominata contenuto file e quindi l'elaborazione riportata nella zona denominata descrizione fasi. Qui è riportato quello che sarà il dettaglio dell'esecuzione e tutti i dettagli del ciclo. Il pulsante TRASFERISCI NEL CUBLOC trasferisce attraverso la porta seriale selezionata i dati elaborati nel CUBLOC. Il pulsante immissione configurazione permette infine la scrittura del file di testo associato al ciclo.

SVILUPPI

In un prossimo articolo studieremo altri cicli pneumatici come i cicli con sequenze in parallelo, cicli con sequenze condizionali e i cicli che prevedono salti.

di ALBERTO TRASIMENI

USARE IL PORT EXPANDER CON I PIC

CONTROLLO VENTOLA DI COOLING DOWN

(parte seconda)

Nei circuiti elettronici di potenza il fattore temperatura influisce fortemente sul funzionamento del circuito, come per esempio, nel caso dei driver di potenza utilizzati per il controllo dei motori stepper bipolari. Si può risolvere il problema aggiungendo ai radiatori tradizionali una ventola di raffreddamento

Il progetto verte sul controllo della temperatura raggiunta dai dispositivi di potenza, facendo uso di un sensore termico, che a contatto con il radiatore, in relazione alla temperatura metta o meno in funzione una ventola di raffreddamento, collocata sul radiatore a cui sono fisicamente in contatto i dispositivi di potenza. Il sensore termico impiegato, per il progetto, è il ds18s20 della Dallas semiconductor. La struttura funzionale del sensore è riportata in figura1.

Figura 1: Struttura funzionale DS18S20

Prima di illustrare il progetto sarà essenziale descrivere il funzionamento del sensore su cui esso si basa. Il ds18s20 e un *sensore* digitale che utilizza il protocollo one-wire per la comunicazione di tipo asincrona con il microcontrollore. Prima di iniziare una trasmissione il microcontrollore ”master” deve inviare il segnale di reset, per cui la linea dati è posta a livello logico “0” per un tempo almeno pari a 480us. Quando si dovrà trasmettere lo “0” logico la linea dati deve restare a livello basso(0v) per un tempo pari a 60us, mentre quando si dovrà trasmettere un “1” la linea dati deve rimanere bassa un tempo almeno pari a 15us. Questo modo di comunicare viene anche definito per Time Slot. Ogni Time Slot deve avere una durata di 60us, nel caso in cui si invia un “1” logico la linea dati resterà a “0” per 15us e si troverà a “1” per un tempo pari a 45us. Infine nel caso venga inviato uno “0” logico la linea dati resterà a livello logico basso per un tempo pari al Time Slot. Tra uno Slot e il successivo deve intercorrere un tempo pari ad 1us(tempo tra bit e bit). Il ds18s20 come

Figura 2: Struttura dello SCRATCHPAD.

quasi tutti i dispositivi one-wire è dotato di scratchpad che rappresenta la memoria RAM del sensore, infatti i dati vengono trasferiti mediante tale memoria che è formata da più registri ad 8 bit, ognuno dei quali svolge una funzione particolare, come mostrato in figura2.
Il valore della temperatura viene memorizzato nel byte0 e byte1 ed in complemento a due se dovesse risultare essere negativa. La temperatura verrà quindi normalmente letta con una risoluzione di 9 bits, cioè è possibile misurare il mezzo grado celsius.
I registri byte2 e byte3 permettono di memorizzare un eventuale temperatura di allarme, qualora non si intenda leggere la temperatura ma stabilire solamente quando questa ha raggiunto un valore prestabilito. I bytes 6 e 7 servono invece per avere una risoluzione maggiore di 9 bit e possono essere sfruttati tramite una relazione fornita dal costruttore e presente sul datasheet. Per poter utilizzare il sensore è necessario inoltre conoscere i comandi che consentono di svolgere le diverse operazioni su di esso. Per una descrizione dettagliata e completa di tali comandi è necessario fare riferimento al datasheet del sensore, qui di seguito riporterò i comandi generalmente più usati (usati nel software di gestione del progetto) ed il loro significato raccolti nella tabella 1.
La Mikroelettronika fornisce il compilatore C dotato di libreria, estremamente semplice e versatile da usare, per la gestione del protocollo one-wire. Qui di seguito riporto mediante le figure 3-4 i listati del programma in C per le operazioni di lettura della temperatura e per la lettura del codice ROM (usati nel progetto).
Un'altra parte importate del progetto è rappresentata dal port expander mcp23s17 di

```
Ow_Reset(&PORTE, 2);              // invio il comando di reset
Ow_Write(&PORTE, 2, 0xCC);        // invio comando SKIP_ROM
Ow_Write(&PORTE, 2, 0x44);        // invio comando CONVERT_T
Delay_us(60);                     // aspetto 60us
Ow_Reset(&PORTE, 2);              // invio comando di reset
Ow_Write(&PORTE, 2, 0xCC);        // invio comando SKIP_ROM
Ow_Write(&PORTE, 2, 0xBE);        // invio comando READ_SCRATCHPAD
temp_int=Ow_Read(&PORTE, 2);      // leggo temperatura(byte0)
temp_fraz=Ow_Read(&PORTE,2);      // leggo temperatura(byte1)
```

Figura 3: Lettura temperatura.

COMANDO	VALORE HEX	SIGNIFICATO
SKIP ROM	0xCC	Si utilizza questo comando quando si vogliono effettuare operazioni senza l'invio del codice ROM.
CONVER T	0x44	Questo comando avvia una singola conversione di temperatura. A seguito della conversione, il valore della temperatura è memorizzato nei registri byte0 e byte1 dello scratchpad e il DS18S20 ritorna al suo stato di inattività a basso consumo.
READ SCRATCHPAD	0xBE	Questo comando permette al master di leggere il contenuto dello scratchpad. Il trasferimento dei dati inizia con il bit meno significativo del byte 0 e continua attraverso lo scratchpad fino al 9 byte (byte 8 - CRC).
WRITE SCRATCHPAD	0x4E	Questo comando permette al master di scrivere 2 byte. Il primo byte è scritto nel registro TH, e il secondo byte viene scritto nel registro TL (temperatura allarme).
READ ROM	0x33	Questo comando permette di leggere il codice ROM

Tabella 1: riassunto dei comandi principali.

```
Ow_Reset(&PORTE, 2);           // RESET del bus onwire
Ow_Write(&PORTE, 2, 0x33);     // leggo la rom del sensore
Delay_us(120);                 // Aspetto 120us
stato=Ow_Read(&PORTE,2);       // Assegno a stato l'indirizzo del sensore
```

Figura 4: Lettura indirizzo sensore termico.

```
//--------------- configuro registro IOCONA-------------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_IOCONA,0x20);
//----------------configuro registro  IOCONB------------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_IOCONB,0x20);
//----------------configuro la porta B come uscita------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_IODIRB,0x03);
//----------------scrivo sulla porta B 0x00-------------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_OLATB,0x00);
//----------------configuro porta A come ingressi-------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_IODIRA,0xE0);
//---------------Configuro interruzione PORTA-A---------------//
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_GPINTENA,0xFF);
   Write_2317(MCP2317_OPCODE_WRITE,MCP2317_INTCONA ,0x00);
```

Figura 5: Funzioni programmazione MCP23S17.

```
unsigned char Read_2317(const unsigned char addressd,const unsigned char addressr)
{
  unsigned char dato;
  CS=0;                    // abilita MCP3217
  SSPBUF=addressd;         // invia OPCODE(lettura)
  while(PIR1.SSPIF==0);
  SSPIF_bit=0;
  SSPBUF=addressr;         // invia indirizzo registro da cui leggere
  while(PIR1.SSPIF==0);
  SSPIF_bit=0;
  SSPBUF=0x00;             // invia dato fittizio per la lettura
  while(PIR1.SSPIF==0);
  SSPIF_bit=0;
  while(SSPSTAT.BF==0);    // si attende fino a che il dato non è presente
  dato=SSPBUF;             // assegna a dato il valore letto
  CS=1;
  return dato;
}
```

Figura 7: Corpo funzione lettura.

cui sono state illustrate le caratteristiche generali, cioè i registri di funzionamento e la loro programmazione, nell'articolo precedente. Ho utilizzato il port expander per interfacciare dei led di segnalazione relativi al valore della temperatura massima ammissibile, al valore normale della temperatura e all' inconveniente che il sensore termico possa rompersi o essere collegato in modo non corretto. Oltre ai led, il port expander gestisce un lcd a caratteri 20x4, presente sulla porta B, con controllore HD44780 della Hitachi. L'emulazione del bus per il display a cristalli liquidi è stata tarata per i prodotti di nuova generazione della Winstar di cui la Mikroelettronika è il rivenditore. Sul display vengono visualizzate le informazioni relative allo stato del sensore, se operativo "On" e qualora questo dovesse non funzionare o essere mal collegato, viene visualizzato il messaggio: "Controllare sensore termico DS18S20". Questo messaggio permane fintantoché il sensore non viene sostituito o collegato in modo corretto. I driver di gestione del port expander sono stati scritti utilizzando i registri della porta SPI, quindi sono facilmente modificabili ed adattabili a qualsiasi altra applicazione, dicasi la stessa cosa per l'emulazione del bus dell'lcd. Prima di poter utilizzare il port expander questo deve essere inizializzato, cioè devono essere programmati i registri delle porte A e B, sia come definizione dei pin(I/O) che nella modalità di funzionamento. Per questo riporto le funzioni che assolvono a tale compito in figura5. La funzione Write_2317() permette di programmare il modo di funzionamento delle porte e di definire i pin di ingresso/uscita per ciascuna porta. Il corpo della funzione viene riportato in figura6.
Dalla figura6 si nota che la prima operazione da effettuare è quella di portare a livello logico basso il chip select del dispositivo e successivamente caricare il registro sspbuf, buffer di trasmissione, con opcode di scrittura. L'istruzione successiva cioè la while, struttura iterativa a condizione iniziale, testa il bit sspif che rappresenta il flag di fine trasmissione, tale flag appartiene a registro PIR1 nel caso del PIC16F887. Infatti il flag viene portato al valore logico "1" quando la trasmissione è stata ultimata, per questo è necessario portarlo di nuovo a livello logico "0" prima di poter effettuare una nuova trasmissione. Successivamente si invia l'indirizzo del

```
void Write_2317(const unsigned char addressd,const unsigned char addressr,unsigned char dato)
 {
   CS=0;                    // abilita MCP3217
   SSPBUF=addressd;         // invia OPCODE (scrittura)
   while(PIR1.SSPIF==0);    // si attende fino a che il buffer non è vuoto
   SSPIF_bit=0;             // cancello flag di fine trasmissione
   SSPBUF=addressr;         // invia indirizzo registro su cui scrivere
   while(PIR1.SSPIF==0);
   SSPIF_bit=0;
   SSPBUF=dato;             // invia il dato da scrivere
   while(PIR1.SSPIF==0);
   SSPIF_bit=0;
   delay_us(300);           // Ritardo stabilizzazione bus mcp23s17
   CS=1;
 }
```

Figura 6: Corpo funzione scrittura.

```
//----------- Configurazione Modulo SPI---------------//
 SSPCON.SSPEN =1;    // Attivo periferica SPI
 SSPCON.CKP    =1;    // Stato Idle clock alto
 SSPCON.SSPM0 =0;    // frequenza clock Fosc/4
 SSPCON.SSPM1 =0;
 SSPCON.SSPM2 =0;
 SSPCON.SSPM3 =0;
 SSPSTAT.SMP = 0;    // Campionamento dati a metà del periodo di clock
 SSPSTAT.CKE = 0;    // dato trasmesso sul fronte salita clock
 SSPSTAT.BF=0;       // Cancello bit buffer full per sicurezza
```

Figura 8: Configurazione della porta SPI.

LCD1
DMC-20434_16

DIN
DOUT
SCK
SS
TRIG
SPI

R4 10k
R1 4.7K
U2
RE3/MCLR/VPP
RA0/AN0/ULPWU/C12IN0-
RA1/AN1/C12IN1-
RA2/AN2/VREF-/CVREF/C2IN+
RA3/AN3/VREF+/C1IN+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RA6/OSC2/CLKOUT
RA7/OSC1/CLKIN
RB0/AN12/INT
RB1/AN10/C12IN3-
RB2/AN8
RB3/AN9/PGM/C12IN2-
RB4/AN11
RB5/AN13/T1G
RB6/ICSPCLK
RB7/ICSPDAT
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/P1A/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0 RD1 RD2 RD3 RD4
RD5/P1B
RD6/P1C
RD7/P1D
RE0/AN5
RE1/AN6
RE2/AN7
PIC16F887

C1 22pf
X1
C3 22pf

R2 10k
R3 18R

U1
SCK SI CS SO RESET A0 A1 A2
GPA0 GPA1 GPA2 GPA3 GPA4 GPA5 GPA6 GPA7
GPB0 GPB1 GPB2 GPB3 GPB4 GPB5 GPB6 GPB7
INTA INTB
MCP23S17
TEMPMAX TEMPOK ALLSENS
RS E D4 D5 D6 D7

RV1 10k
R5 220R
R6 220R
D1 D2
TEMPOK TEMPMAX

+12V
R13 680
D5 1N4007
D6
RL1 G5C-1-DC12
R8 4.7k
C2 100nF
U4 4N25
R10 330
R12 2.2k
R11 10k
Q1 BC337
Volts
SW1 SW-SPST
U3
VCC DQ GND
24.5
DS18S20
R9 220R
R7 220R
D3 D4
ALLSENS
CLK CE RST

A B C D

labcenter Electronics
PROTEUS
The Complete Electronics Design System

TITLE: Sistema Controllo Ventola
DATE: 14/06/13
PAGE: 1/1
BY: Ing.Trasimeni
REV: 1/1
isis ARES

Figura 9: Schema elettrico del progetto.

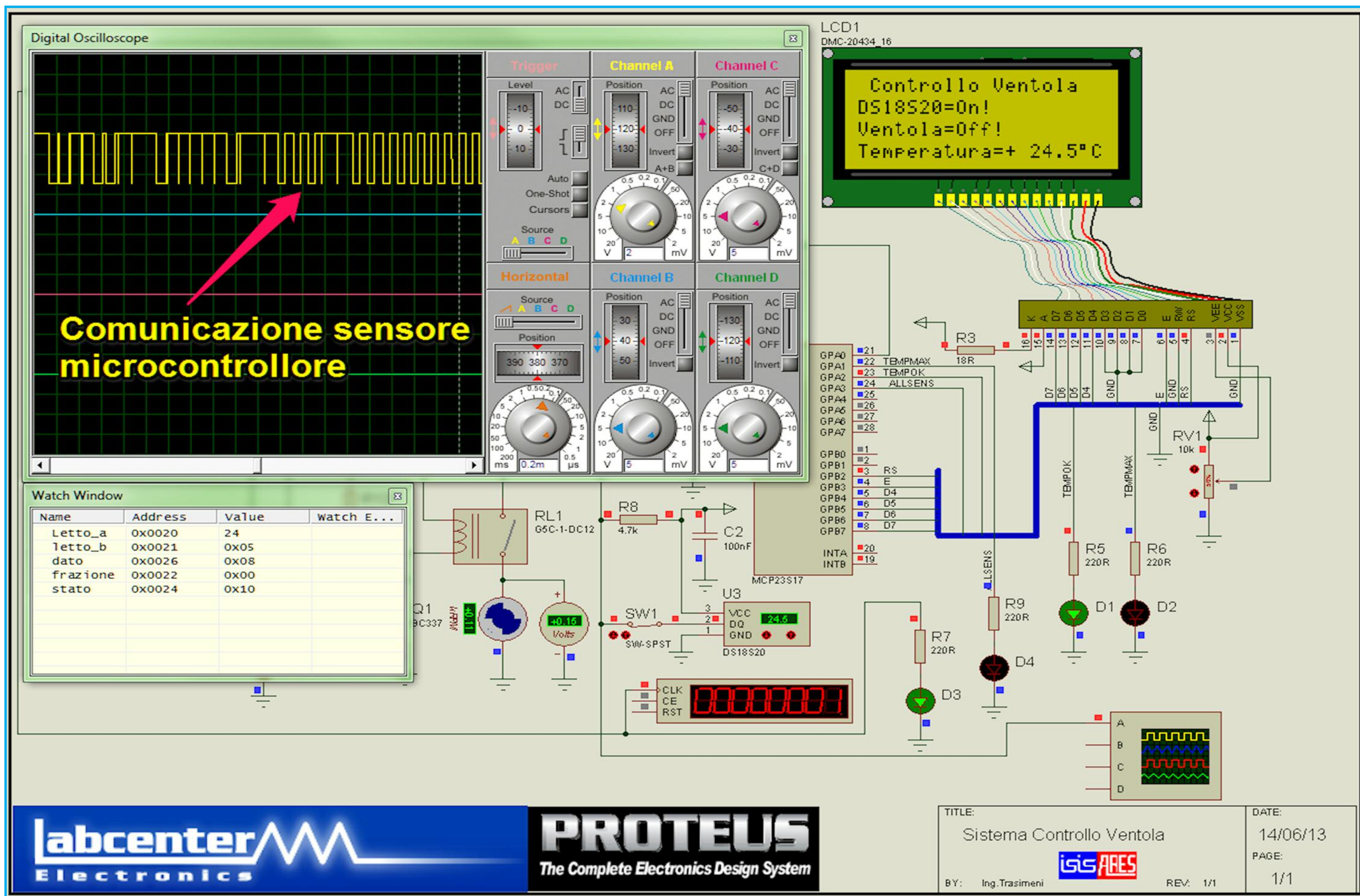



Figura 10: Simulazione con protocollo Time Slot.

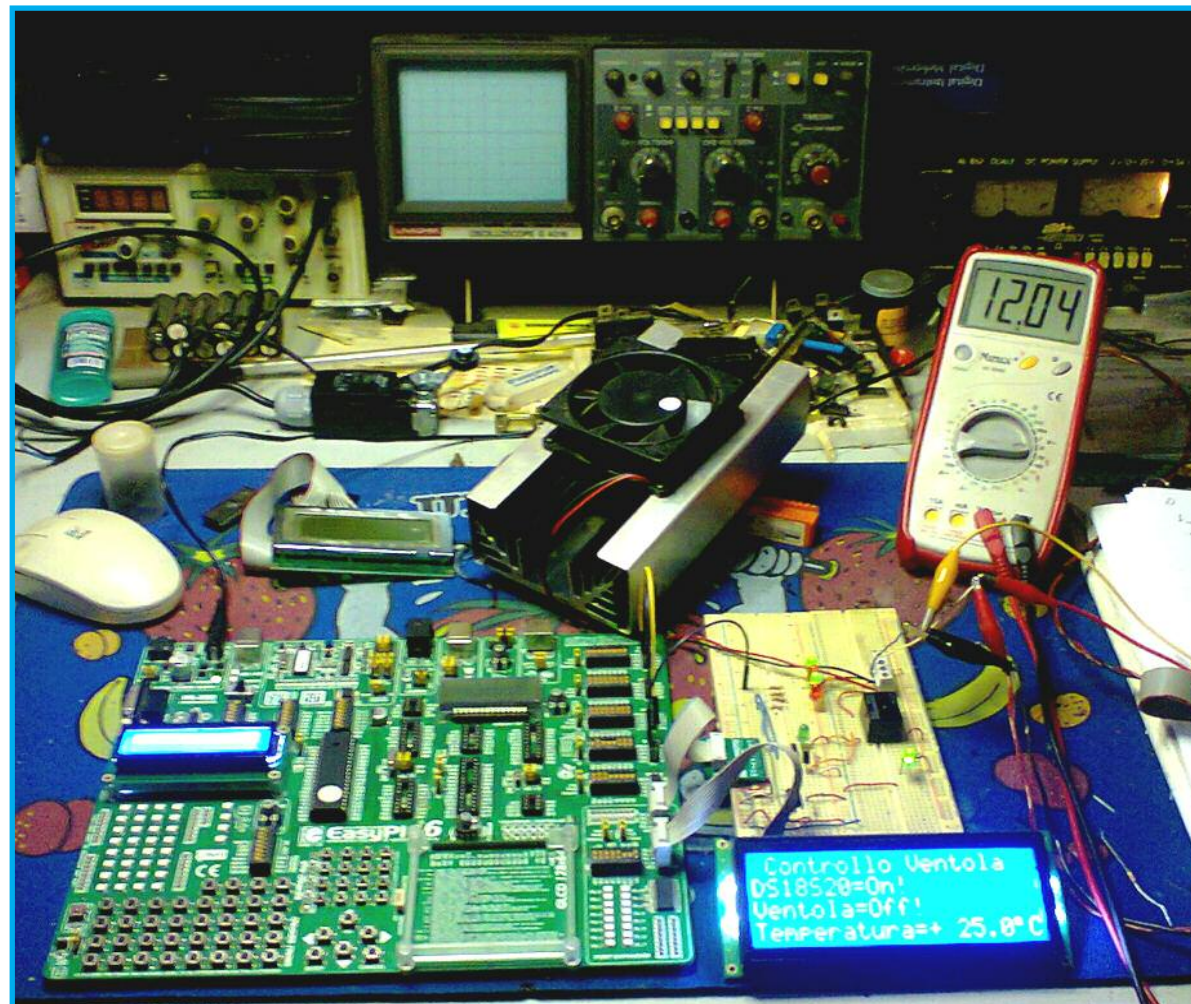



Foto 1: Ventola a riposo.

Foto 2: Allarme sensore.

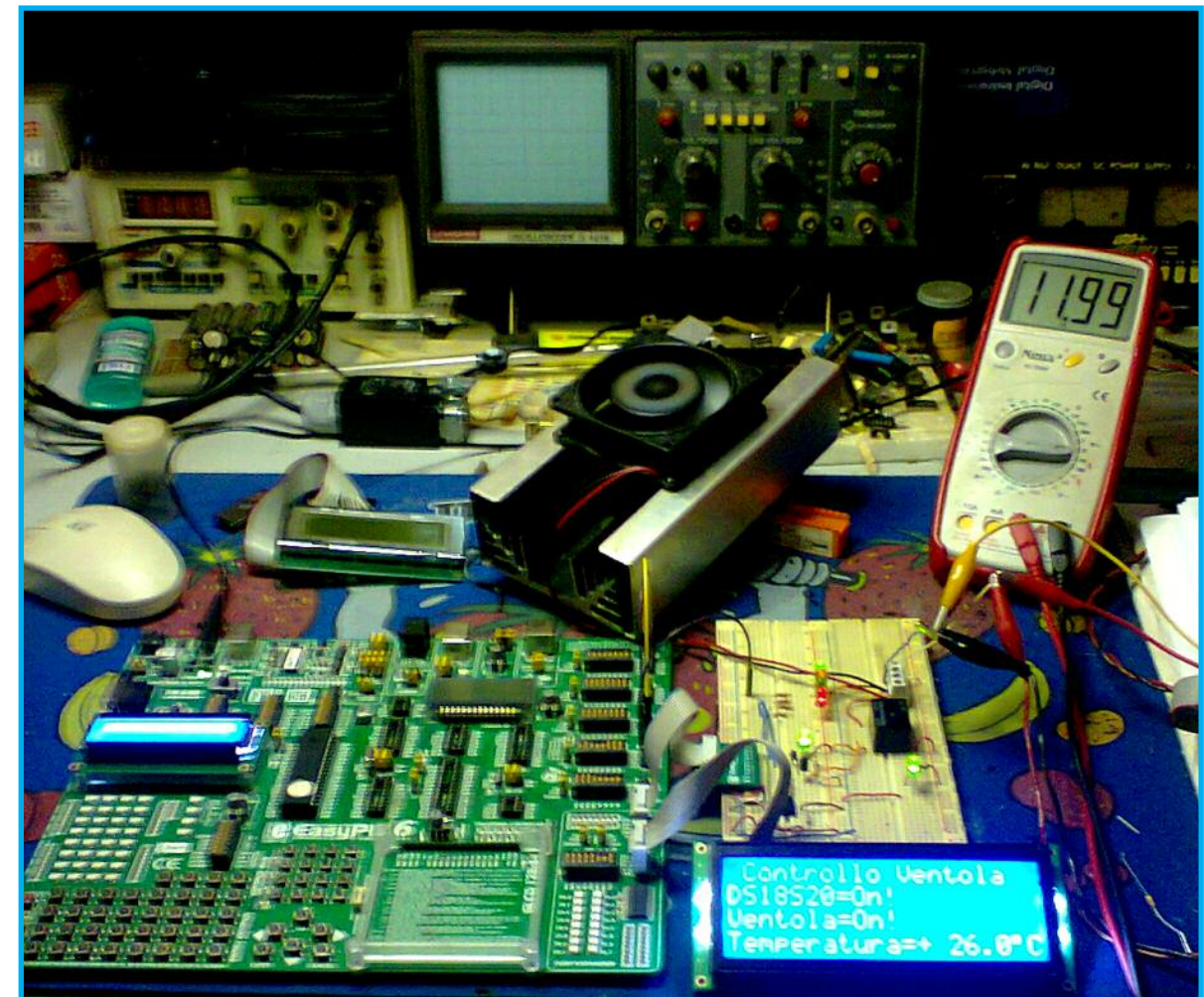



Foto 3: Ventola in funzione.

registro su cui si vuole effettuare l'operazione di scrittura e per ultimo il dato, che può rappresentare una parola di controllo o un dato vero e proprio. Prima della fine dell'operzione di scrittura ho inserito una routine di ritardo per stabilizzare il bus del mcp23s17, la scrittura termina riportando a livello logico "1" il chip select. Una seconda importante funzione di interfaccia per mpc23s17 è quella di lettura dei registri che riporto in figura7.

Anche nel caso della lettura la prima operazione da effettuare è portare a livello logico basso il chip select. Si scrive in sequenza opcode di lettura e l'indirizzo del registro da cui si vuole leggere l'informazione, il fatto nuovo in questo caso, è che per leggere il dato dal registro desiderato, bisogna inviare prima della lettura, un dato fittizio, in modo tale che lo shift register dello slave(mcp23s17) possa fare uscire il dato, che sarà caricato nel registro sspbuf del master(microcontrollore).

La corretta operazione di lettura è stabilita dal bit bf(buffer full status bit) che dovrà essere a livello logico "1" affinché la ricezione sia stata completata correttamente. Il bit bf appartiene ad uno dei registri della SPI ed in modo particolare al registro di stato sspstat. In questo caso la while(ricezione completata), come mostra la figura7, dovrà testare tale bit. Vediamo ora come configurare la porta SPI del microcontrollore(PIC16F887), la configurazione viene mostrata in figura8.

La configurazione della porta SPI avviene utilizzando due registri, uno di controllo sspcon ed uno di stato sspstat. Mediante i bit sspen, ckp, sspm0, sspm1, sspm2, ssmp3 del registro sspcon è possibile, at-

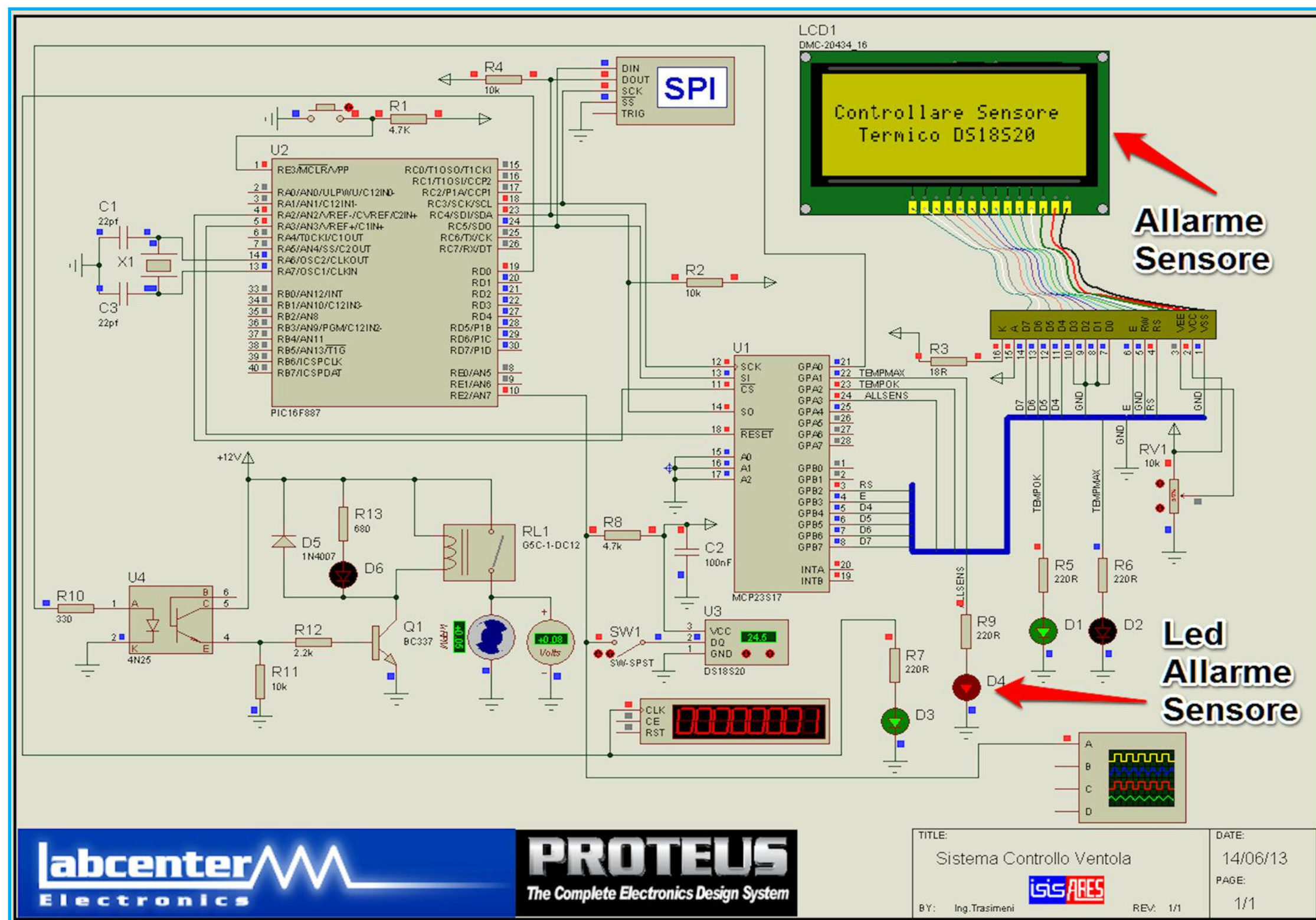



Figura 11: Simulazione con allarme sensore.

tivare la porta SPI, definire lo stato logico a riposo della linea del clock ed il rapporto tra la frequenza dell'oscillatore esterno, generalmente un quarzo, e la frequenza del clock utilizzato dalla porta SPI. Nel nostro caso, questo rapporto vale 1/4, cioè la frequenza del clock della porta SPI è la quarta parte di quella dell'oscillatore esterno. I bit smp, cke del registro di stato sspstat, permettono di stabilire dove avverrà il campionamento dei bit costituenti il dato(a metà del periodo del clock nel nostro caso) e su quale fronte del segnale di clock i bit del dato saranno trasmessi(fronte di salita nel nostro caso). Questi bit di configurazione della porta SPI devono essere opportunamente definiti in relazione alla periferica con cui la porta SPI deve comunicare. Lo schema elettrico del progetto è rappresentato in figura9.

Dallo schema elettrico si nota la presenza del fotoaccoppiatore 4n25 il cui scopo è quello di isolare galvanicamente le alimentazioni, 5volts con cui lavora tutta la parte digitale del sistema, da quella dei 12volts

Elenco componenti

R1	10 KΩ 1/4 W
C1	22pf ceramico
C2	22pf ceramico
X1…	quarzo 8mhz
4N25	fotoaccoppiatore
R10	330Ω 1/4 W
R11	10 KΩ 1/4 W
R12	2. 2 KΩ 1/4 W
R13	680Ω 1/4 W
R8	4. 7 KΩ 1/4 W
R7	220Ω 1/4 W
R9	220 Ω 1/4 W
R5	220Ω 1/4 W
R6	220Ω 1/4 W
R3	18 Ω 1/4 W
R2	10 KΩ 1/4 W
mcp23s17	port expander (Microchip)
ds18s20	sensore termico (Dallas)
D1	led 5mm(rosso)
D2	led 5mm(verde)
D4	led 5mm(rosso)
D3	led 5mm(verde)
D6	led 5mm(verde)
D5	1n4007(diodo)
PIC16F887	Microcontollore (Microchip)
Ventola	12Volts (secondo esigenze di progetto)
lcd 20x4	Winstar
G5C-1-DC12	relè (Ormon) 12v

di funzionamento della ventola. La resistenza R10 in ingresso al fotoaccoppiatore serve per limitare la corrente di funzionamento del fotodiodo di ingresso del 4n25. Il suo valore è stato scelto in modo tale da

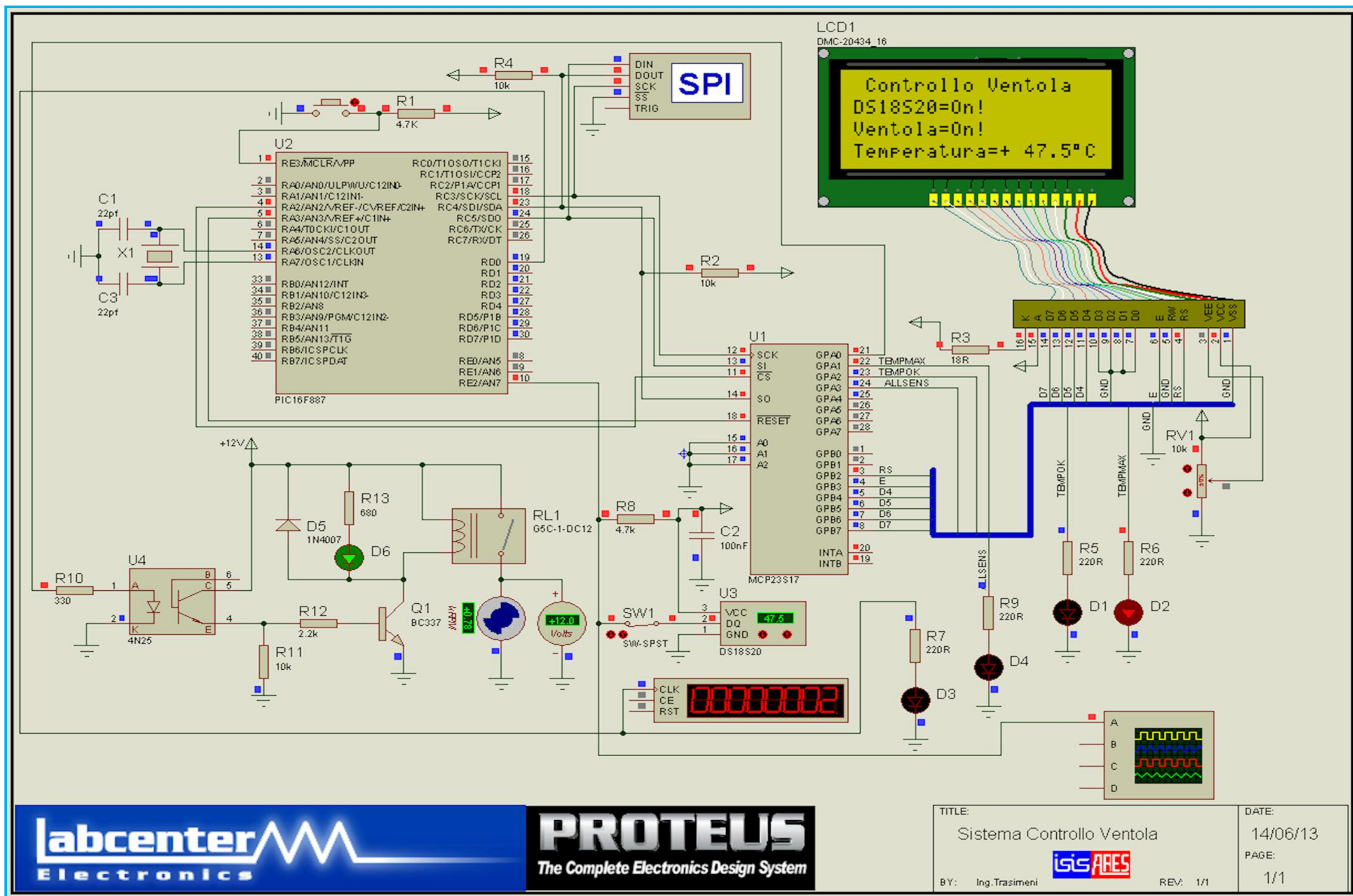



Figura 12: Simulazione con ventola in funzione.

garantire la saturazione del transistor Q1(bc337), che eccita il relè di azionamento della ventola, e nel contempo da non caricare il pin GPA0 del port expander a cui è collegata. I diodi d1 e d2, segnalano con la loro accensione lo stato di temperatura normale o del raggiungimento della temperatura massima, a cui la ventola deve entrare in funzione.

Il diodo d4 viene acceso solo quando il ds18s20 è stato collegato non correttamente o nella peggiore delle ipotesi quando questo si è rotto. Il diodo d3 lampeggia con una cadenza di 500ms per segnalare lo stato di attività corretta da parte del sistema. Infine il diodo d6 si accende quando la ventola entra in funzione.

La resistenza R8, collegata alla linea dati del sensore, rappresenta una resistenza di pull-up in quanto l'uscita del sensore è una uscita open-drain, senza tale resistenza non ci potrebbe essere nessuna comunicazione tra sensore e microcontrollore. Il condensatore C2 da 100nf permette di filtrare l'alimentazione del sensore dal rumore presente su tale linea e quindi di garantirne il corretto funzionamento.

Il progetto è stato inoltre simulato mediante il programma Proteus 7. 10 e i risultati della simulazione vengono mostrati con le figure 10-11-12. Lo schema elettrico di figura9 risulta conforme al cablaggio realizzato sulla easy-pic 6 della Mikroelettronika, come mostrano le foto successive, è perfettamente funzionante.

Le foto 1, 2 e 3 mostrano una temperatura di intervento della ventola di 26 °celsius, questo perché toccando il sensore potessi avviarla. Nel caso reale tale temperatura dovrà essere tra 45° e 50 ° gradi celsius.

OSSERVAZIONI FINALI

Il progetto poteva essere realizzato senza far uso del port expander, ma facendo un po' di conti, i pin impegnati del microcontrollore sarebbero stati dieci, con l'uso del port expander solo cinque e avendo ancora a disposizione ben nove pin sulle porte A e B del dispositivo. Siccome volevo mostrarne un esempio di tipo applicativo ne ho colto l'occasione. Certamente usando un microcontrollore che possiede un numero limitato di pin, l'uso del port expander sarebbe stato obbligatorio. Per quanto riguarda il sensore termico si sarebbe potuto utilizzare senza far uso della linea di alimentazione, cioè sfruttando la cosiddetta connessione parassita, fornendo quindi l'energia al dispositivo mediante lo stesso segnale presente sulla linea dati, questo grazie alla capacità C_{pp} di ingresso del sensore come mostrato in figura1.

Clicca qui per scaricare il software in mikroC e il file di simulazione:

Download

di Massimiliano Anticoli

BENVENUTO ARDUINO ROBOT!

Un vero robot? Si, proprio un vero Robot su ruote! Il primo prodotto ufficiale Arduino che si estende al di là delle celeberrime schede con microcontrollore!

Presentato ufficialmente al Maker Faire di San Mateo, Arduino Robot è stato progettato assieme a Com-publot, vincitore per ben 4 volte del “Robocup Junior robotics soccer”, ovvero il campionato di calcio per robot.
Il robot è già completamente assemblato e pronto per essere utilizzato senza alcuna saldatura!

figura 1: il kit Arduino Robot

E’ sufficiente collegare lo schermo LCD a colori, caricare le (quattro) batterie incluse, lanciare l’Arduino IDE e caricare il codice di esempio, incluso, tramite il cavo USB.
Arduino Robot è dotato di ingressi integrati in gran numero: due potenziometri, cinque bottoni, una bussola digitale, cinque sensori di piano, e un lettore di schede SD. Inoltre, ha anche un altoparlante e uscita per schermo a colori e molto spazio per la prototipazione e connettori pre-saldati per espansione TinkerKit: in maniera tale da personalizzare il robot con elettronica e sensori aggiuntivi!
Arduno Robot, come per altri Kit Arduino, viene fornito con undici progetti passo – passo, per poter iniziare a conoscere la potentissima piattaforma robotica (per chi magari “mastica poco” di elettronica e meccanica).

ASPETTO ESTERIORE

Arduino Robot è composto da due schede circolari posta una sopra l’altra, con due ruote. Ogni scheda circolare ha circa 19 cm di diametro, mentre l’altezza totale

figura 2: vista da sotto

(comprese di ruote, schermo GTFT e altri connettori), è di circa 10 cm.
La scheda superiore (Control Board) contiene display LCD a colori, uno slot per schede microSD, una EEPROM, un altoparlante, una bussola, una manopola più alcuni tasti e LED.
La scheda inferiore (Motor Board) è la sede di quattro batterie AA (NiMH), una coppia di motori e ruote, un connettore di alimentazione e l'interruttore, più alcuni sensori a infrarossi.

ALL'INTERNO

Arduino Robot è dotato di due processori (ATmega32u4 di Atmel collegate via cavo – 16 Mhz – 32KB Flash Memory), uno su ciascuno delle schede circolari. La Motor Board controlla i motori (scheda inferiore)e la Control Board (scheda superiore) legge i sensori e decide come operare, oltre al controllo della visualizzazione. Ciascuna dei due pannelli circolari è una scheda Arduino vera e propria, completa e programmabile utilizzando l'IDE di Arduino.

Il robot ha molti dei suoi pin mappati come sensori di bordo e attuatori.
La programmazione del robot è simile ad Arduino Leonardo (avendo a bordo gli stessi microprocessori). Entrambi i processori sono dotati di comunicazione USB, eliminando la necessità di un processore secondario. Questo permette al robot di apparire a un computer collegato come una porta seriale /COM virtuale (CDC).
Come per tutte le schede Arduino, ogni elemento della piattaforma: hardware, software e documentazione è open-source., dunque è possibile implementare le caratteristiche grazie alla grande community internazionale.

ALIMENTAZIONE

Arduino robot può essere alimentato tramite il 4 batterie ricaricabili NiMH AA (è specificato di non utilizzare battere non ricaricabili) o con collegamento USB. La fonte di alimentazione viene selezionata automaticamente.
Per motivi di sicurezza, i motori sono disabilitati quando il robot è alimentato tramite connessione USB.
Arduino Robot è fornito con un caricabatterie che richiede 9V di alimentazione esterna proveniente da un adattatore AC-DC. Il caricabatterie non funziona se alimentato da USB.
La Control Board è invece alimentata dalla tensione di alimentazione della Motor Board.

MEMORIA

L' ATmega32u4 dispone di 32 KB (con 4 KB utilizzati per il bootloader), ha anche

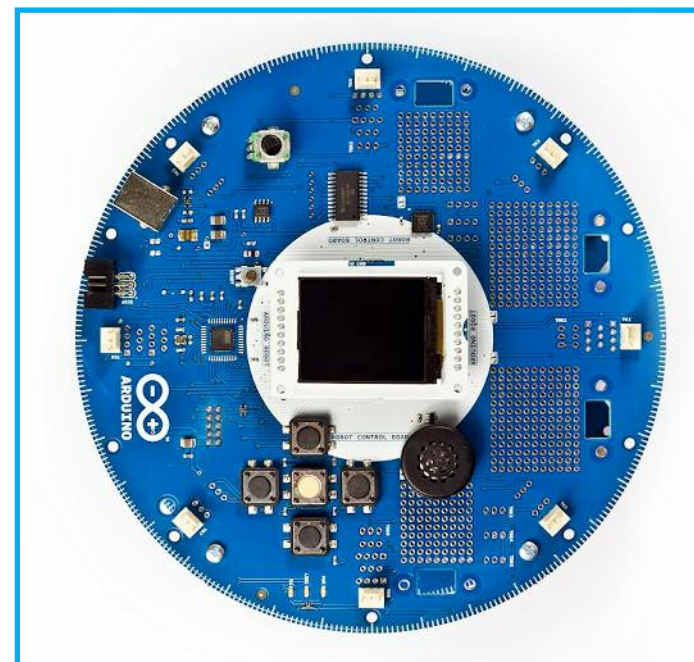



figura 3: vista dall'alto

2,5 KB di SRAM e 1 KB di EEPROM (che può essere letta e scritta con la libreria EEPROM).
La Control Board ha 512 Kbit EEPROM extra che può essere letta tramite I2C.
E' incluso anche un lettore di schede SD esterna collegata allo schermo GTFT da cui si può accedere dal processore della Control Board per lo storage.

COMUNICAZIONE

Arduino Robot può facilmente comunicare con un computer, con un altro Arduino o altri microcontrollori. L'ATmega32U4 fornisce la comunicazione seriale UART TTL (5V).
Come già detto, l'ATmega32U4 consente la comunicazione seriale (CDC) via USB e appare come una porta COM virtuale per il software sul computer.
Il chip funziona anche come dispositivo a piena velocità USB 2.0, utilizzando i driver COM USB standard.
Ognuna delle due schede ha un identificatore di prodotto USB separato e apparirà come porte differenti sull'IDE. L'ATmega32U4 supporta anche la comunicazione I2C e SPI. Il software Arduino include una libreria Wire per semplificare l'uso del bus I2C, per la comunicazione SPI, si utilizza, invece, la libreria SPI.

PROGRAMMAZIONE

Il robot può essere programmato con il software Arduino scaricabile dal sito ufficiale Arduino (vedere tabella).
I processori ATmega32U4 sull'Arduino Robot vengono precaricati con un bootloader che permette di caricare un nuovo codice ad esso senza l'uso di un programmatore hardware esterno. Esso comunica utilizzando il protocollo AVR109.
È anche possibile ignorare il bootloader e programmare il microcontrollore attraverso l'header l'ICSP (In-Circuit Serial Programming).
Ognuna delle due schede, come accennato, ha un ATmega32U4, che può essere programmato indipendentemente dall'altro.

figura 4: la confezione di Arduino Robot

E' consigliabile per i principianti programmare la Control Board, per prendere familiarità con le funzionalità del Robot, per poi passare alla programmazione della Motor Board.

CONTENUTO DELLA SCHEDA SD

La scheda SD viene utilizzata per memorizzare le informazioni da far leggere al robot.
LA SD Card viene fornita precaricata con immagini e file audio, utilizzati negli esempi. Una volta conosciute le varie funzionalità del robot, si dovrebbe essere in grado di aggiungere i propri file.

PRONTI ALL'USO?

Quando si apre il robot, è necessario impostare un paio di cose prima che sia pronto a partire, dopodiché una applicazione precaricata che vi chiederà il vostro nome… e a questo punto sarete pronti per iniziare il viaggio nella robotica open source grazie ad Arduino!

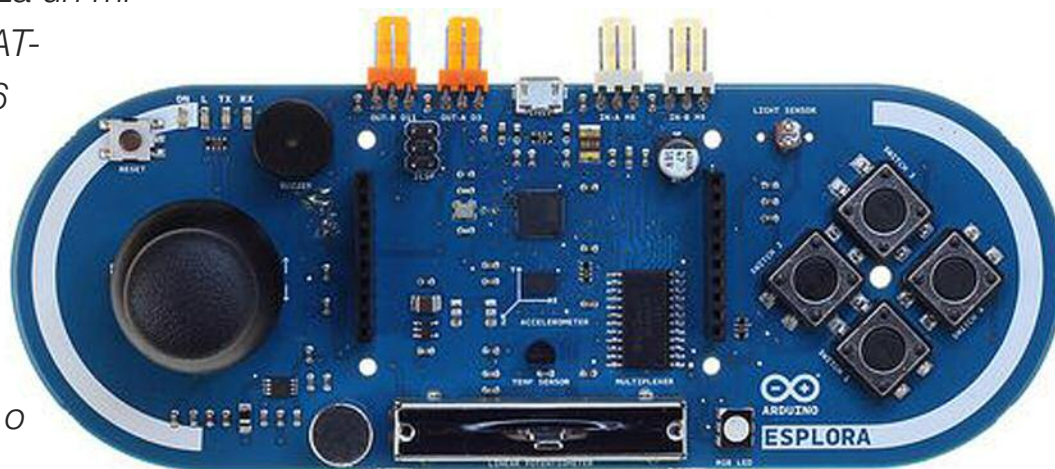
Arduino Esplora: entra nel mondo dell'elettronica senza conoscerla!

La nuova scheda Arduino Esplora è progettata per gli utenti che vogliono iniziare a lavorare con Arduino, senza dover conoscere l'elettronica!

Arduino Esplora è una scheda derivata da Arduino Leonardo e si differenzia da tutte le schede Arduino precedenti in quanto fornisce una serie di sensori a bordo per l'interazione.

Arduino Esplora ha uscite audio integrate, diversi sensori di input, un joystick, un cursore, un sensore di temperatura, un accelerometro, un microfono e un sensore di luce. E'anche possibile espandere le sue capacità grazie ai due ingressi Tinkerkit e connettori di uscita, e di un connettore per uno schermo LCD a colori TFT.

Arduino Esplora utilizza un microcontrollore AVR ATMEGA32U4 con 16 MHz oscillatore a cristallo e di una connessione micro USB in grado di agire come un dispositivo client USB, come un mouse o una tastiera.

Per approfondire

[1] http://youtu.be/UwqH9atAHns - Video: cosa contiene il Kit
[2] http://arduino.cc/en/Main/Robot - Tutta la documentazione ufficiale di Arduino Robot
[2] http://arduino.cc/en/Guide/Robot - Guida ad Arduino Robot

di GIUSEPPE LA ROSA

QUATTRO RELÈ VIA SERIALE O BLUETOOTH

Il progetto descritto in questo articolo mostra come realizzare una scheda capace di pilotare fino a quattro relè tramite porta seriale o wireless mediante Bluetooth

La scheda (**figura 1**) può controllare fino a quattro carichi ad esempio lampade o piccoli elettrodomestici con un assorbimento massimo di 5 ampere.
Essa può essere gestita tramite collegamento seriale al PC e relativo software per Windows, può anche essere controllata con il Bluetooth, con l'aggiunta del modulo Bluetooth HC-06 che permette il controllo wireless sia tramite PC o Smartphone Android.

Figura 1: Foto della scheda a 4 relè controllata tramite seriale.

IL CIRCUITO ELETTRICO

In **figura 2** è illustrato lo schema elettrico della scheda seriale.
I quattro relè (K1 a K4) vengono gestiti da Arduino UNO, un ATMEGA328P (IC2), ed il circuito che ne risulta è molto semplice, come si evince dalla **figura 2**.
Poiché i relè (K1 a K4) non possono essere connessi direttamente alle porte del microcontrollore IC2, perché hanno un assorbimento elevato maggiore della corrente massima erogabile dal microcontrollore, per ovviare a questo problema sono stati usati i transistor T1 a T4 in configurazione d'interruttore elettronico con i relativi diodi Damper D2 a D4 che agiscono da soppressori di sovratensioni, impedendo alle extra tensioni generate dalle bobine dei relè di attraversare i quattro transistor.
Il circuito è predisposto per il collegamen-

Figura 2: Schema elettrico della scheda a 4 relè controllata tramite seriale.

to alla porta seriale del PC dalla quale riceve i comandi per l'accensione o spegnimento dei relè mediante il connettore seriale X1.

La scheda, deve essere alimentata con un alimentatore stabilizzato da 12 volt in grado di erogare una corrente di 1,5 ampere. Il connettore J2 serve per la programmazione In Circuit ed è utile per caricare il Bootloader di Arduino UNO. L'ATMEGA328P deve solo interpretare i comandi in standard RS232 e generare i

Elenco componenti

R1÷R2	470 Ω 1/4 W
R3	10 KΩ 1/4 W
R4	470 Ω 1/4 W
R5÷R7	12 KΩ 1/4 W
R8÷R11	470 Ω 1/4 W
R12	12 KΩ 1/4 W
C1	100 µF 35 V elettrolitico
C2	100 nF poliestere
C3	100 µF 35 V elettrolitico
C4	100 nF poliestere
C5	100 µF 35 V elettrolitico
C6	22 pF ceramico
C7÷ C8	100 nF poliestere
C9	22 pF ceramico
C10	10 µF 100 V elettrolitico
C11	100 µF 35 V elettrolitico
C12÷C14	10 µF 100 V elettrolitico
D1÷D5	1N4007 diodo
T1÷T4	BC337
IC1	HIN232CPZ
IC2	ATMEGA328P
IC3	L7805CV
LED1	Led 5 mm Arancio
LED2	Led 5 mm Verde
LED3	Led 5 mm Giallo
LED4÷LED7	Led 5 mm Rosso
Y1	Quarzo 16 MHz
K1÷K4	Relé 12 V/6 A
S1	Pulsante c.s.
J1	Presa DC 90° 5,5 x 2,1 mm
J2	Strip maschio 3+3 pin
X1	Connettore DB9 femmina 90°
X2	Morsetti 8 Poli
P1	Ponticello cavo rigido
N.1	Vite 3M più dado
N.1	Zoccolo 14+14 pin
N.1	Zoccolo a tulipano 8+8 pin
JP1*	Strip maschio 8 pin torniti
JP2*	Strip femmina 4 pin
N.1*	Modulo HC-06

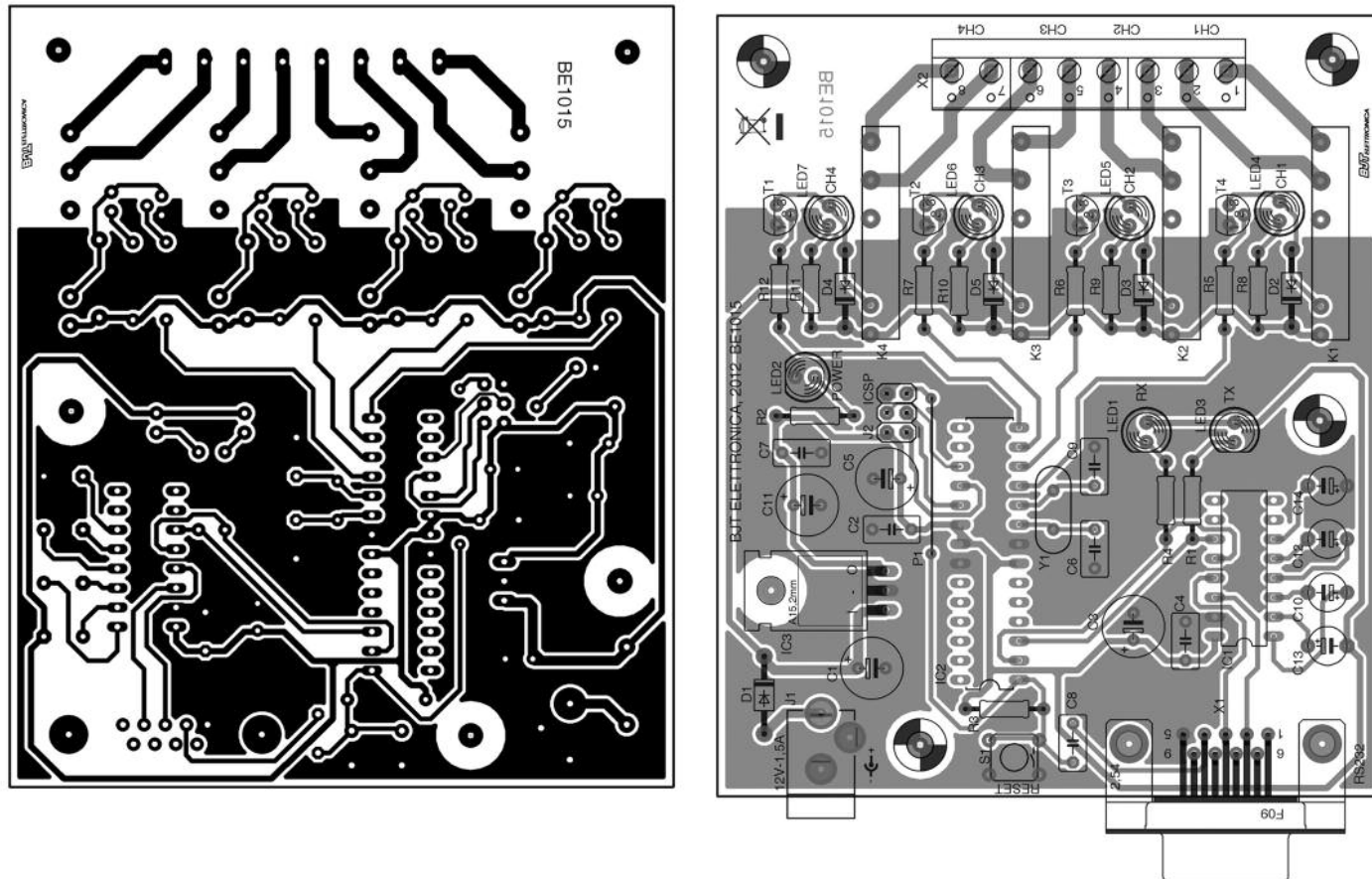



Figura 3: Master lato saldature e piano di montaggio.

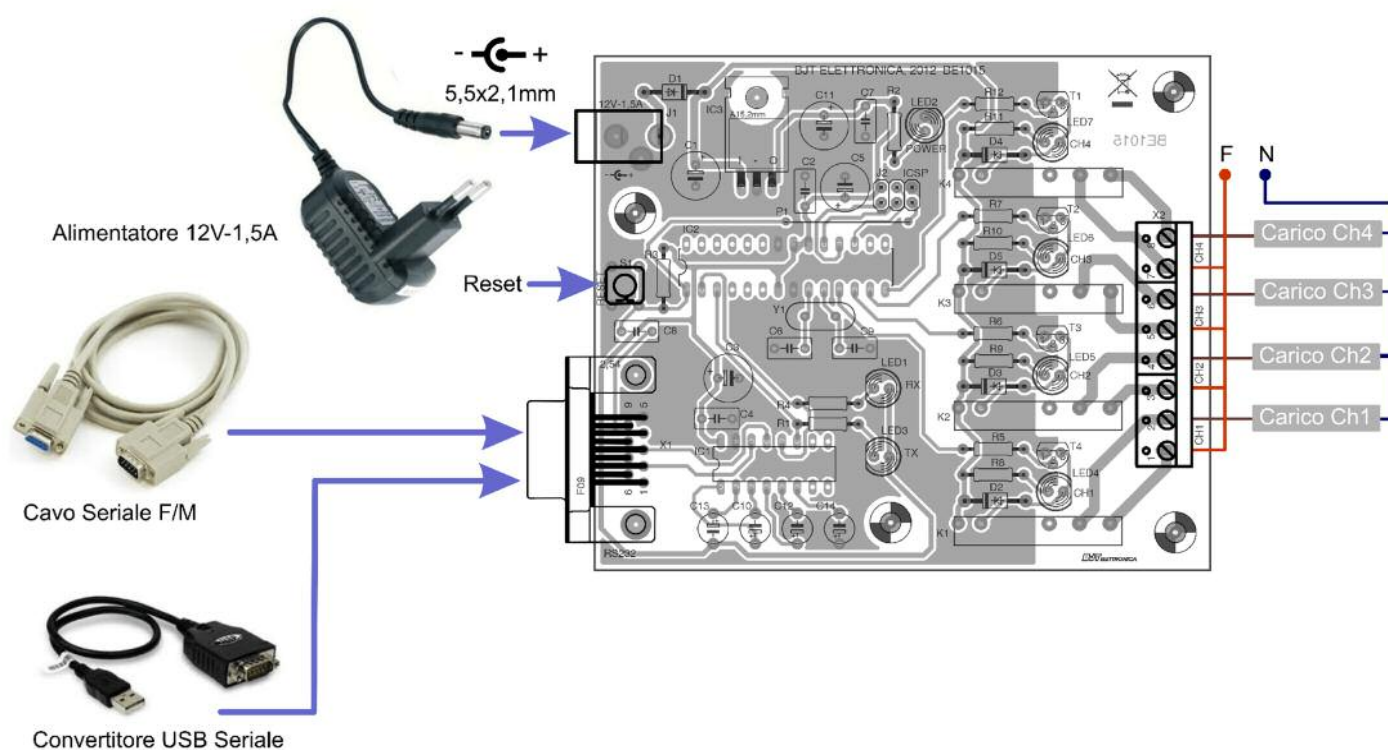



Figura 4: Schema di collegamento.

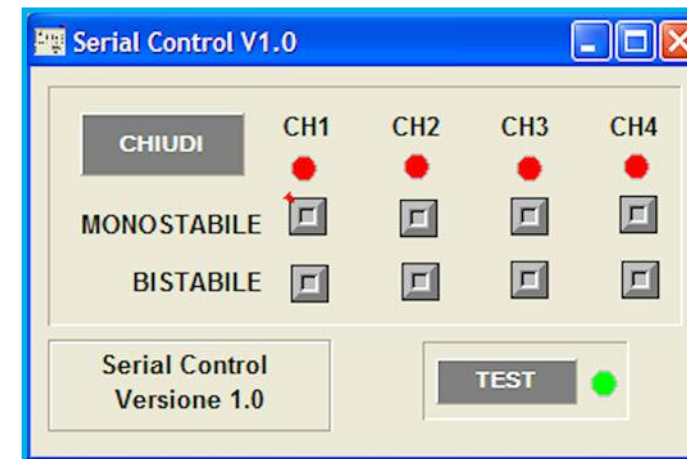



Figura 5: Schermata principale del Software di controllo.

segnali elettrici che controllano i relè secondo il formato 0 e 5 volt.

Il circuito integrato IC1 è un classico circuito di interfaccia RS232/TTL.

Con soli quattro condensatori elettrolitici esterni (C10, C12, C13 e C14) IC1 è alimentato a 5 volt ed è in grado di generare le tensioni di ±10 volt compatibili con lo standard RS232 e di convertire i dati con livelli TTL (livelli 0 e 5 volt) in dati RS232 e viceversa. IC1 interfaccia il microcontrollore IC2 con il PC, tramite il connettore seriale X1.

Il modulo Bluetooth HC-06 si connette al-

HARDWARE - Blocco note

File Modifica Formato Visualizza ?

```
[1.cvs1]
Name=Serial interface
COM=1
[2.cvs1]
Name=Serial interface
COM=1
[2.cvs2]
Name=Serial interface
COM=1
[2.cvs3]
Name=Serial interface
COM=1
[2.cvs4]
Name=Serial interface
COM=1
[2.cvs5]
Name=Serial interface
COM=1
[2.cvs6]
Name=Serial interface
COM=1
[2.cvs7]
Name=Serial interface
COM=1
[2.cvs8]
Name=Serial interface
COM=1
[2.cvs9]
Name=Serial interface
COM=1
[2.cvs10]
Name=Serial interface
COM=1
```

Figura 6: File d'inizializzazione "HARDWARE.INI".

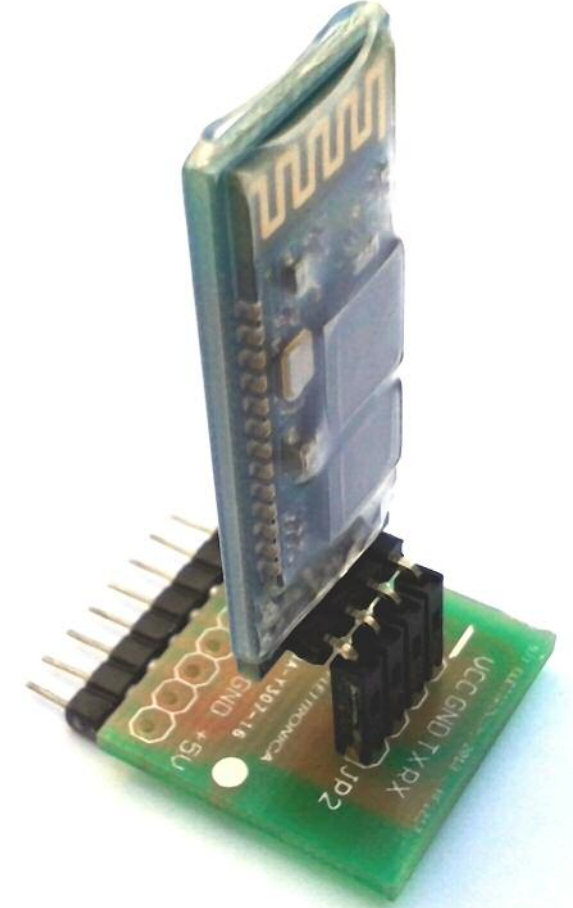



Figura 7: Modulo HC-06 montato sulla scheda adattatrice.

la scheda seriale tramite lo zoccolo dell'integrato IC1 e ai pin 9 a 16, con l'ausilio della scheda adattatrice di **figura 8**, nella figura è rappresentato lo schema elettrico, il layout del PCB e il master.

Essa ha il compito di adattare i pin dell'integrato IC1 alla disposizione dei pin del modulo HC-06.

Nel prossimo paragrafo spiegheremo l'utilizzo del modulo HC-06

IL MODULO BLUETOOTH HC-06

Il protocollo Bluetooth è nato per sostituire la linea seriale cablata.

Si può utilizzare per stabilire una connessione senza fili tra Microcontrollori e PC anche Smartphone e altri dispositivi, su modello di configurazione Master/Slave.

A differenza del modello HC-05, che può

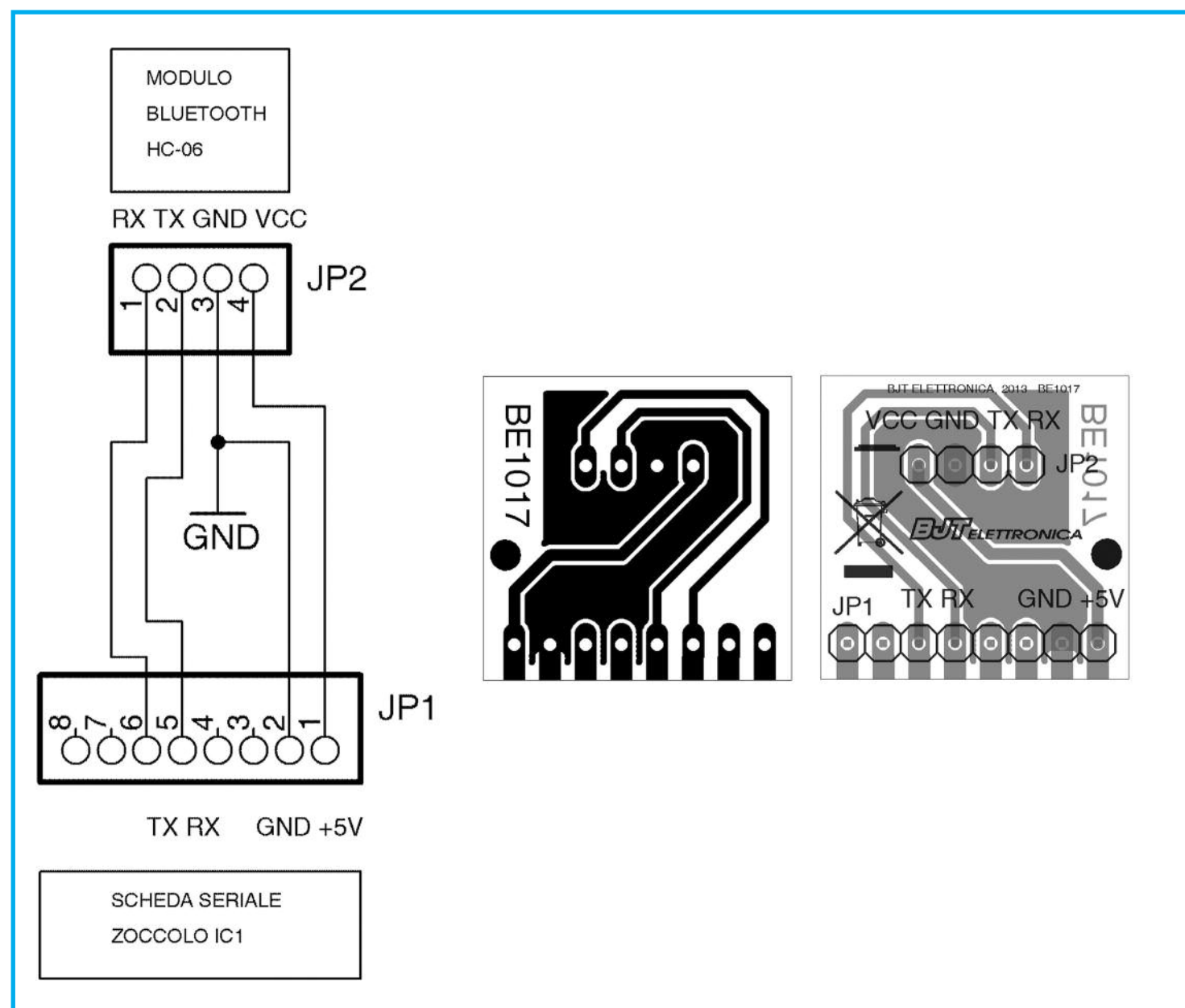



Figura 8: Schema elettrico, master lato saldature e piano di montaggio della scheda adattatrice.

Figura 10: Software Hercules per la programmazione del modulo HC-06.

essere settato dall'utente come Master o Slave, il modello HC-06 (**figura 7**) è possibile acquistarlo come Master o Slave, e non può essere modificato dal'utente.

Esso supporta i comandi AT per modificare velocità di trasmissione, nome dispositivo, passkey, ha un settaggio predefinito della porta seriale "8-N-1 9600" cioè dati a 8 bit, nessuna parità, 1 bit di stop con velocità 9600 Bps.

La passkey predefinita di accoppiamento è la seguente "1234".

Per il nostro progetto imposteremo la porta seriale a "8-N-1 19200" la passkey a "1234" e il nome come "Serial Control".

Avvalendoci di convertitore USB a seriale TTL, ed eseguendo i collegamenti al modulo HC-06 come illustrato in **figura 9** possiamo modificare tutti i parametri appena descritti.

Utilizzando il software Hercules (**figura 10**) andremo ad inviare per prima il comando "AT" che ci serve a verificare il collegamento e la porta seriale sia stata selezionata correttamente.

Se il modulo risponde "OK" significa che tutto è stato connesso correttamente è possiamo proseguire con l'invio degli altri comandi.

Inviamo il comando per cambiare il nome del dispositivo con questo comando "AT+NAMESerial Control", poi il comando per cambiare la passkey "AT+PIN1234", ed in fine il comando per cambiare la velocità della porta a 19200 Bps con il comando "AT+BAUD5".

Figura 9: Collegamento per la programmazione del modulo HC-06.

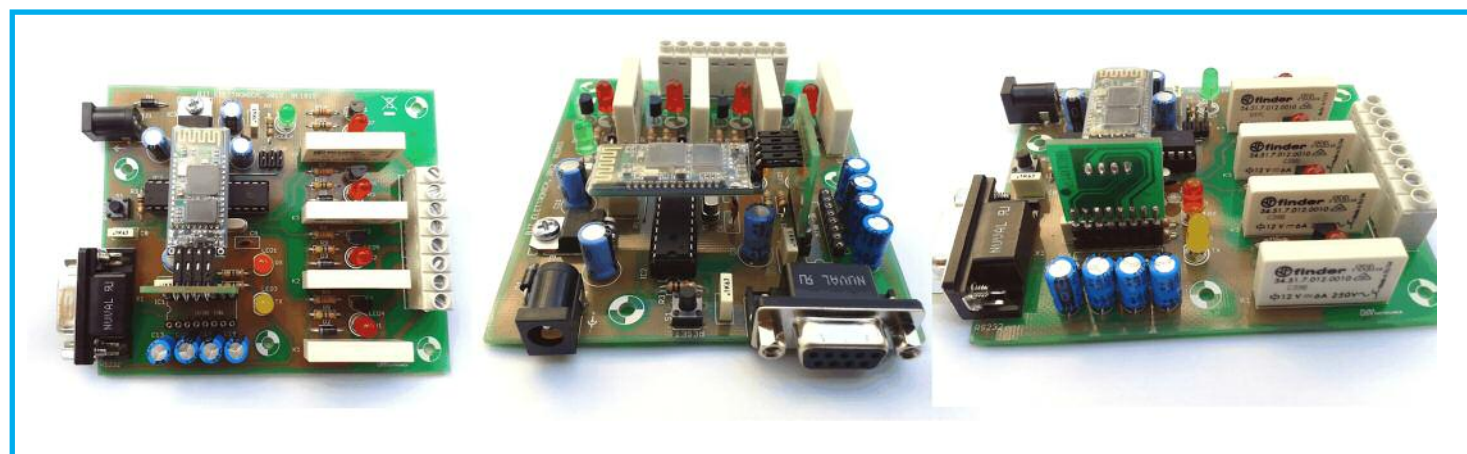



Figura 11: Dettagli per l'installazione del modulo bluetooth sulla scheda seriale.

Con questa semplice procedura abbiamo settato il modulo e adesso è pronto per essere utilizzato.
Per approfondire gli altri comandi consiglio di leggere il Datasheet.
Il modulo va connesso prima alla scheda adattatrice e poi alla scheda seriale in **figura 11** mostra alcuni dettagli per l'installazione.

IL FIRMWARE

Il firmware dell'ATMEGA328P è stato scritto in linguaggio C (Arduino UNO).
Il sorgente del firmware è disponibile per il download sul sito della rivista ed possibile eseguire tutte le modifiche che si vogliono.

Comandi seriali

	ON	OFF	Monostabile
CH1 (relè K1)	2	1	A
CH2 (relè K2)	4	3	B
CH3 (relè K3)	6	5	C
CH4 (relè K4)	8	7	D

Bps 19200 8-N-1

Tabella 1: comandi supportati dalla scheda.

Il firmware riconosce la sequenza di caratteri riportata in **tabella 1** "Comandi seriali" come si può vedere nelle tre colonne "ON", "OFF" e "Monostabile" per ogni canale ci sono tre caratteri due per attivare i relè in modo bistabile "ON" e "OFF" e il terzo per attivarlo in modo monostabile.
La trasmissione seriale implementata del tipo 8-N-1 cioè dati a 8 bit, nessuna parità, 1 bit di stop con velocità 19200 Bps.

REALIZZAZIONE DELLE SCHEDE E COLLAUDO

Passiamo adesso alla costruzione della scheda che si presenta abbastanza semplice, la basetta è del tipo mono faccia e si prepara a partire della traccia di **figura 3**.
Ottenuto il circuito stampato montate (seguendo il piano di montaggio **figura 3**) i componenti richiesti dalla "**Lista componenti**".
I componenti contrassegnati con l'asterisco vanno montati sulla scheda adattatrice.
Montate i componenti partendo dal ponticello P1, poi inserite resistenze, in seguito diodi e gli zoccoli per gli integrati (lo zoccolo IC1 deve essere a tulipano 8+8 pin), quindi proseguite prima con i condensatori non polarizzati e poi gli elettrolitici, e in fine con il quarzo Y1, lo strip J2, il pulsante S1, i LED, e i connettori X1, X2, J1 e per ultimi i relè.
Per quando riguarda la scheda adattatrice bisogna saldare gli strip maschi torniti sul lato saldature del connettore JP1, mentre lo strip femmina JP2 normalmente sul lato componenti come visibile in **figura 7**.
Se si dispone di un ATMEGA328P già con Bootloader caricato si può passare al caricamento del firmware tramite la porta seriale presente sulla scheda.
Eseguite il collegamento come è rappresentato in **figura 4**, collegate un alimentatore a 12 volt in grado di erogare 1,5 ampere.
A questo punto potete collegare la scheda alla seriale del PC o tramite convertitore USB a seriale, avviate il software per impartire i primi comandi di accensione e spegnimento dei relè.

SOFTWARE PC E L'APPLICAZIONE ANDROID

Prima cosa dovete installare il software "Serial Control V1.0" scaricabile gratuitamente dal nostro sito insieme a tutti gli altri file necessari per la realizzazione di questo progetto.
Durante l'installazione verrà creata un'icona sul desktop per facilitare l'avvio del software la cui schermata iniziale si presenta come visibile in **figura 5**.
Effettuata l'installazione, per prima cosa è

Figura 12: Screenshot dell'App per Smartphone Android e QR Code per il download.

necessario impostate la porta seriale di comunicazione utilizzata poi dovete modificare il file d'inizializzazione "HARDWARE.INI" (vedi **figura 4**) che si trova a questo percorso C:\Programmi\Serial Control V1.0\Settings\HARDWARE.INI, dovete cambiare tutte le "COM=1" con l'indirizzo della porta seriale che state usando nel vostro PC, salvate il file così modificato.

Se usate il Bluetooth dopo aver effettuato l'accoppiamento dovete mettere l'indirizzo della porta che il Bluetooth sta usando. In **figura 5** possiamo vedere la schermata principale del programma, i comandi a disposizione prevedono l'accensione e spegnimento in modo bistabile e in modo monostabile. Per inviare un comando è sufficiente cliccare su uno dei pulsanti.

Il pulsante TEST permette di accendere in sequenza i relè. L'applicazione Android per Smartphone è scaricabile qui o tramite il QR code di **figura 12**. Una volta scaricata e installata nel vostro Smartphone lo Screenshot che vi si presenta è visibile in **figura 12**. Prima di procedere al suo utilizzo bisogna accendere il Bluetooth nello Smartphone ed effettuare la ricerca della scheda che avrà questo nome "Serial Control" dopo di che basta digitare passkey "1234" e la scheda viene accoppiata allo Smartphone. Ora è possibile usare l'applicazione, basta toccare il pulsante "Lista Dispositivi" selezionare "Serial Control", poi toccare "Connetti BT" attendere pochi secondi e lo Smartphone è connesso alla scheda.

Il led sul modulo HC-06 non lampeggerà più rimanendo sempre acceso a testimonianza della connessione avvenuta con lo Smartphone. Come si può vedere dalla **figura 12** anche l'applicazione Android prevede l'accensione e spegnimento in modo bistabile e in modo monostabile, basta toccare uno di questi pulsanti per attivare un relè ed eseguire il comando desiderato.

CONCLUSIONE

Come avete letto in precedenza le applicazioni non si limitano solo a quelle descritte, ma spaziano dall'automazione cancelli alla domotica.

Il sorgente del firmware è scritto in linguaggio C per Arduino UNO ed è facilmente modificabile e adattabile alle proprie esigenze. Il software è stato sviluppato con "ProfiLab-Expert 4.0" mentre l'applicazione Android è stata sviluppata con "App inventor" entrambe adeguabili alle proprie necessità.

Clicca qui per scaricare i file utili alla realizzazione:

Download

Guarda il video della scheda in funzione:

Video

di GIOVANNI DI MARIA

GENERATORE DI ONDA QUADRA

Realizziamo assieme un semplice generatore di segnale ad onda quadra in bassa frequenza, un utilissimo strumento per il nostro laboratorio

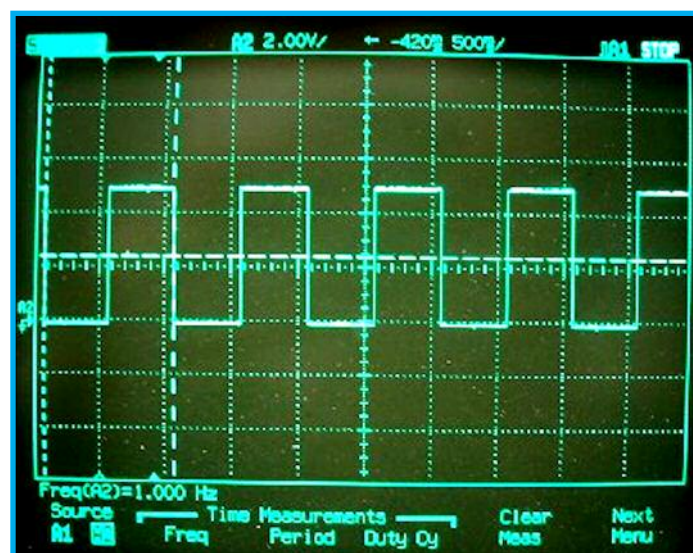



Figura 1: Onda quadra.

In laboratorio non può mancare assolutamente un generatore di onda quadra. Specialmente, se esso è realizzato in proprio, spendendo poco, la soddisfazione è garantita. La generazione del segnale d'uscita avviene senza il bisogno di averne uno in ingresso. Questo strumento è utile in mille occasioni. Con esso è possibile verificare la frequenza di lampeggio con dei diodi Led, oppure può costituire un utile segnale da applicare all'ingresso di altri dispositivi, oppure ancora per provare gli ingressi cadenzati di un microcontrollore o per generare una piccola portante RF (naturalmente con armoniche) o, ancora, per provare contatori digitali.

CARATTERISTICHE PRINCIPALI

Il circuito è caratterizzato da alcune particolarità che lo rendono interessante e utile. A grandi linee ecco le principali peculiarità:

- Tensione d'uscita del segnale: 6V;
- Duty cycle: circa il 50%;
- 8 bande di frequenza selezionabili;
- Regolazione manuale grossa, media e fine;
- Spia luminosa di funzionamento;
- Intervallo globale di frequenza: da < 1 Hz a > 7 Mhz (circa):
- Tensione di alimentazione: 9V minimo.

SCHEMA ELETTRICO

Lo schema elettrico, visibile in figura, è abbastanza semplice. La generazione del segnale, e quindi l'innesco dell'oscillazione, è dovuta al fatto che l'uscita dell'inverter è riportata al suo ingresso, con un opportuno ritardo (o meglio, gradualità) grazie alla presenza del condensatore. Se l'uscita ha un valore logico basso, il suo reinserimento nel ramo di ingresso ne "Inverte" lo stato logico, commutandolo in alto. Il processo si ripete all'infinito. Come si può notare nel grafico delle tensioni presenti all'ingresso e all'uscita dell'inverter, vi sono due situazioni diverse:

- All'ingresso dell'inverter c'è una tensione in salita e discesa, risultato ottenuto dagli effetti della costante di tempo tra

Figura 2: L'oscillatore ad onda quadra.

resistenza e condensatore. Tale tensione, se non si trova nell'intervallo della "indecisione" logica della porta, risulta invertita logicamente in uscita.

- All'uscita dell'inverter troviamo una tensione ad onda quadra abbastanza squadrata. Conviene sempre far seguire questo stadio con un ulteriore inverter che serva da separatore e rinforzi il segnale, se richiesto.

Esaminiamo adesso lo schema completo. L'intero circuito è alimentato dalla tensione stabilizzata da U1, un regolatore 7806 (che fornisce 6V). Potrebbe essere sostituito con un 7805 se occorre in uscita un segnale quadro di 5V oppure con un 7808, se occorrono 8V. In questo ultimo caso l'integrato è ai limiti del suo funzionamento: una tensione maggiore lo distruggerebbe immediatamente!

I condensatori presenti all'ingresso e all'uscita di U1 ne livellano perfettamente la tensione ed evitano eventuali insorgenze di autoscillazioni indesiderate. Il diodo Led D1 testimonia il funzionamento del circuito. La resistenza R4 limita la corrente del diodo a circa 10mA.

L'oscillatore vero e proprio è composto dalla porta inverter U2A. Essa oscilla grazie al componente resistivo presente tra ingresso e uscita e al componente capacitivo presente tra ingresso e massa. Grazie alla esistenza di più trimmer, collegati in serie, e più condensatori (selezionabili mediante un dip switch), la frequenza può essere cambiata entro un vasto intervallo, sufficiente per la maggior parte delle esigenze di laboratorio.

Il gruppo di cinque inverter, formato da U2B, U2C, U2D, U2E ed U2F serve, come detto prima, a separare l'uscita dell'oscillatore e ad irrobustire il segnale in corrente.

La rete in uscita composta da resistenze e condensatori elimina qualsiasi residuo di altissima frequenza spuria, presente eventualmente nel circuito.

Figura 3: Tensioni presenti all'ingresso e all'uscita dell'inverter.

IL CIRCUITO STAMPATO

Compreso il funzionamento del circuito, siamo adesso pronti per approntare il circuito stampato. Dal momento che esso presenta numerose piste, si consiglia di prepararlo con la tecnica della fotoincisione o quella del ferro da stiro. Se si decide di utilizzare la fotoincisione, occorre procurarsi una basetta presensibilizzata e tagliarla nelle giuste dimensioni. Si stampi quindi su lucido l'immagine delle piste di rame, cercando di imprimere quanto più inchiostro possibile. Si impressioni quindi la basetta sotto i raggi ultravioletti per qualche minuto e si sviluppi infine in bagno di acqua e soda caustica. Alla fine si rimuova il rame in eccesso in un bagno di percloruro ferrico. Per maggiori informazioni sulla fotoincisione si possono consultare tanti articoli di progetti precedenti di Fare Elettronica, che trattano in modo approfondito della procedura.

Una volta che il circuito stampato è pronto, occorre forarlo con un piccolo trapano a velocità molto elevata (>8000 g/m) con una punta da 0,8mm per le piazzole dei componenti e con una punta da 1mm per i morsetti da c.s.

Le basette presensibilizzate sono spesso costituite di vetronite, la quale consuma le punte del trapano molto velocemente.

Elenco componenti

R1,R4	1 KΩ 1/4 W
R2	10 KΩ 1/4 W
R3	Ω 1/4 W
C1, C6	100 µF 25 V elettrolitico
C2,C3,C9	100 nF
C4,C7	10 µF 25 V elettrolitico
C5,C10	10 nF
C8	1 µF 25 V elettrolitico
C11	1 nF
C12	100 pF ceramico
C13	10 pF ceramico
C14	1 pF ceramico
U1	Regolatore 7806 (o 7805 o 7808) vedi testo
U2	74HC14 con relativo zoccoletto
D1	Diodo Led Verde
DSW1	Dip Switch con 8 interruttori
J1, J2	Morsetti a 2 posti da c.s.
RV1	Trimmer 22 KΩ
RV2	Trimmer 1 KΩ
RV3	Trimmer 100 Ω
	Cavo schermato
	Contenitore
	Presa BNC femmina

E' opportuno assicurarsi che la continuità elettrica delle piste sia perfetta. Con una lente di ingrandimento (almeno 8x) si controllino accuratamente i dettagli e con un tester, in modalità "prova continuità" si verifichi la bontà dei contatti.

MONTAGGIO E ASSEMBLAGGIO

A questo punto è possibile saldare i componenti elettronici. Con un saldatore di piccola potenza si inizi a lavorare su quelli a basso profilo, per poi continuare con quelli di dimensioni maggiori. Inizialmente si saldino le resistenze, i condensatori cera-

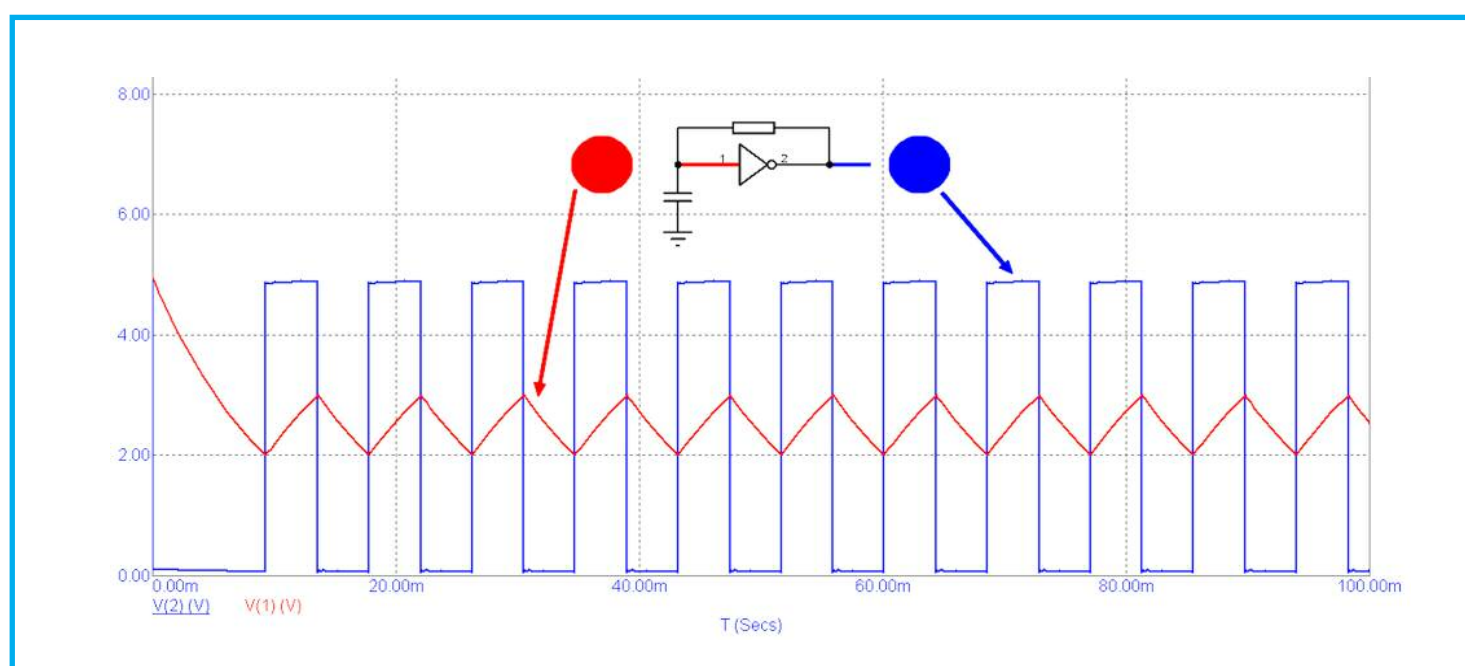
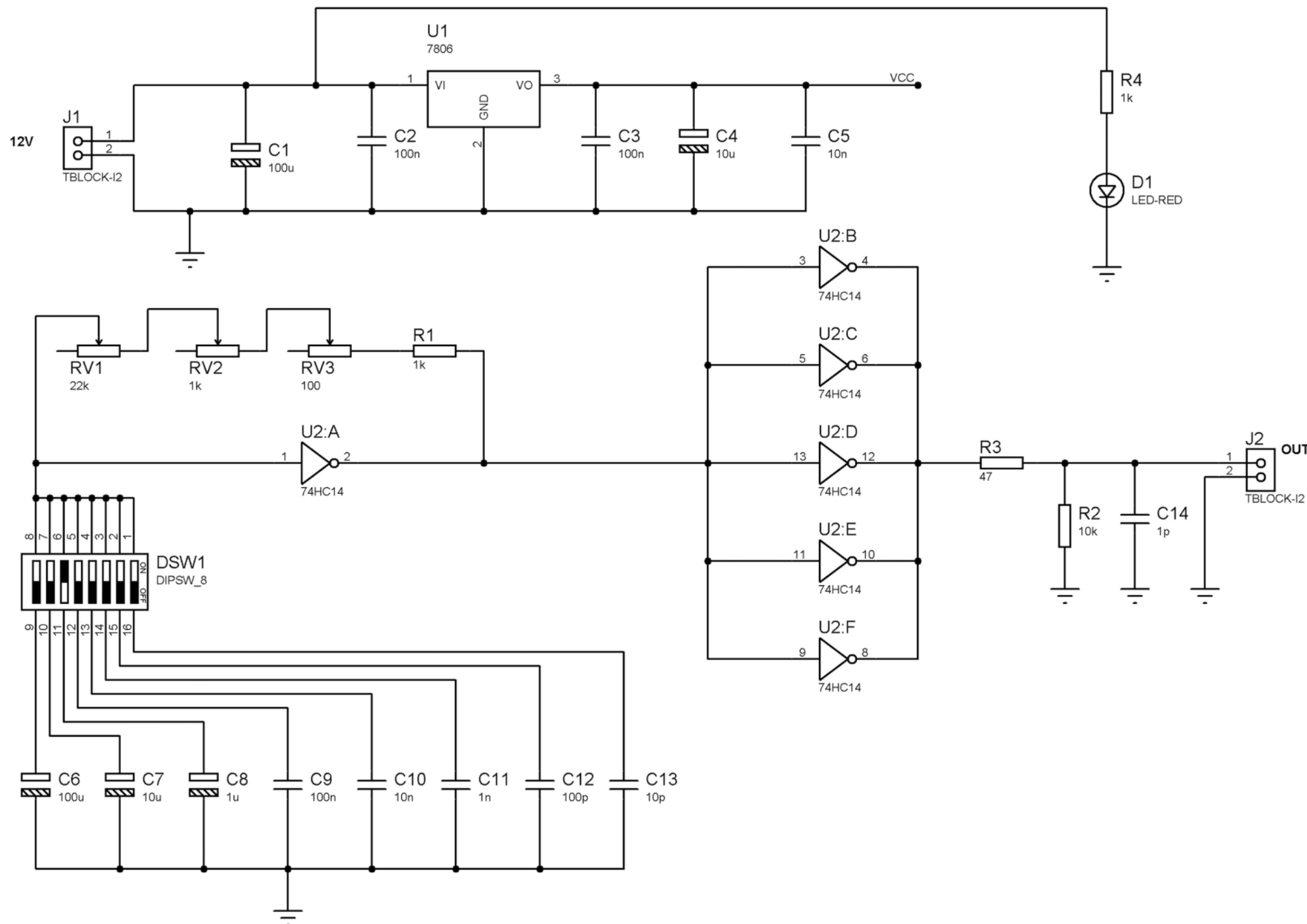



Figura 4: Schema elettrico.

mici, lo zoccolo dell'integrato, e i trimmer. Si passi poi ai morsetti, al diodo Led, al Dip Switch, agli elettrolitici e al regolatore.
Occorre saldare con molta precisione e velocità, evitando di surriscaldare gli stessi. Si verifichi sempre le l'orientamento dei componenti polarizzati con molta attenzione.

COME SI USA

La prima operazione da effettuare è naturalmente quella del collegamento del circuito all'alimentazione. Si puo' utilizzare tranquillamente una sorgente compresa tra 9V e 15V continua, da connettere al connettore contrassegnato con J1. Il diodo Led dovrebbe accendersi testimoniando la corretta alimentazione del dispositivo. Si può quindi collegare al connettore J2 (l'uscita del circuito) il circuito o l'utilizzatore da pilotare. Può essere un ingresso di un PIC, un diodo Led, un oscilloscopio o un frequenzimetro (o addirittura un altoparlante).
Quindi si posizionino tutti i dip switch in posizione OFF e si attivi solo il primo (posto più a sinistra). Esso seleziona la frequenza più bassa. Ricordiamo che deve essere attivato un solo interruttore alla volta.
Si regolino quindi tutti i trimmer (o potenziometri) per il minimo della loro corsa. Si verificherà che il trimmer RV1 regola la frequenza molto grossolanamente, il trimmer RV2 la regola con un passo medio ed il trimmer RV3 con una risoluzione molto fine.

Figura 5: Pinout regolatore 780x.

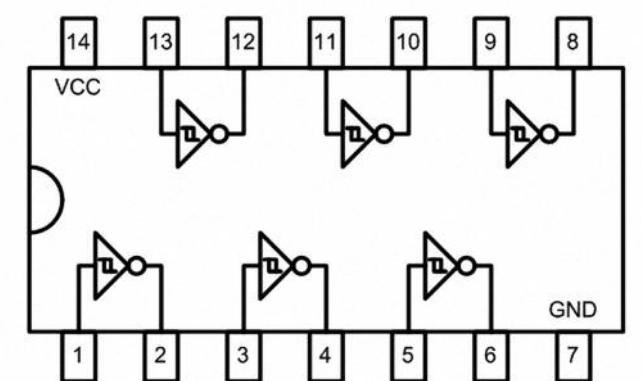



Figura 6: Pinout 74HC14.

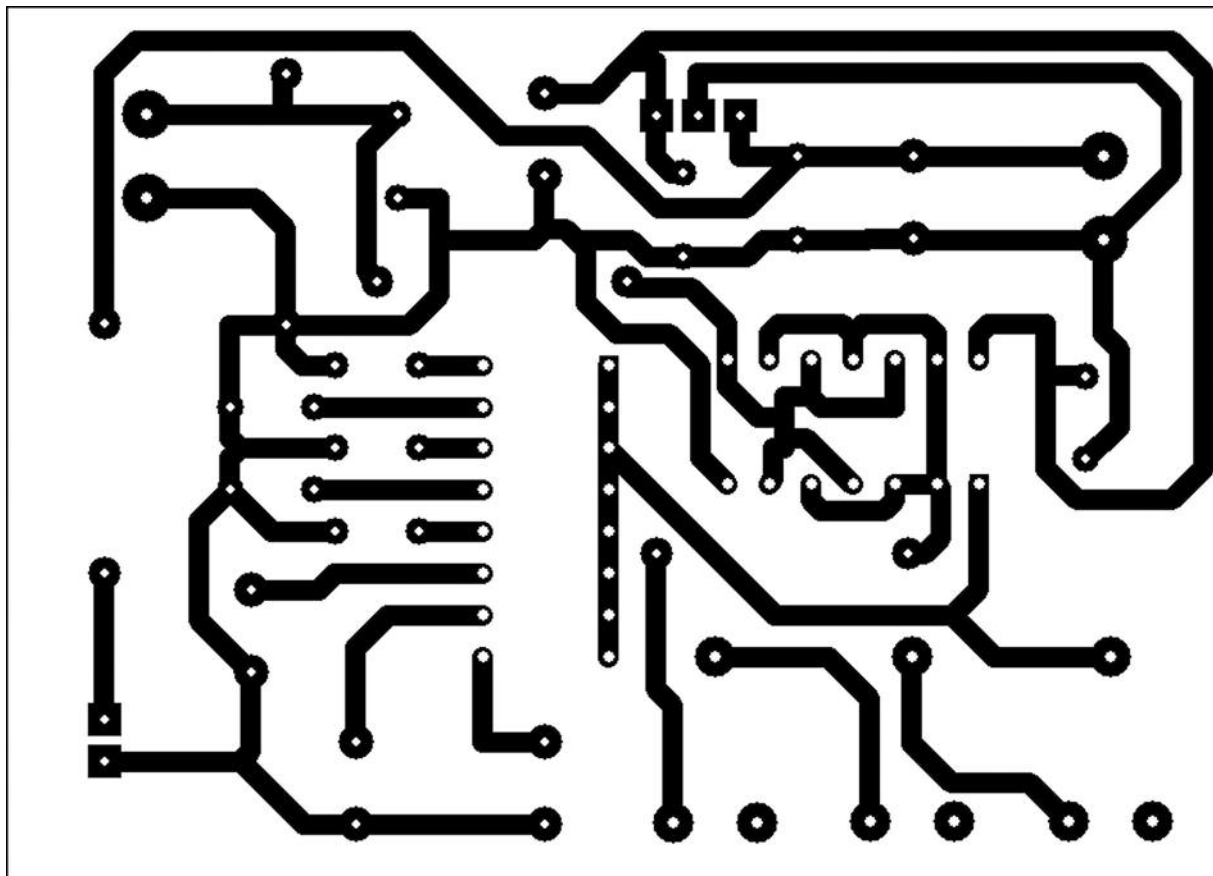



Figura 7: Il PBC lato rame in scala 1:1

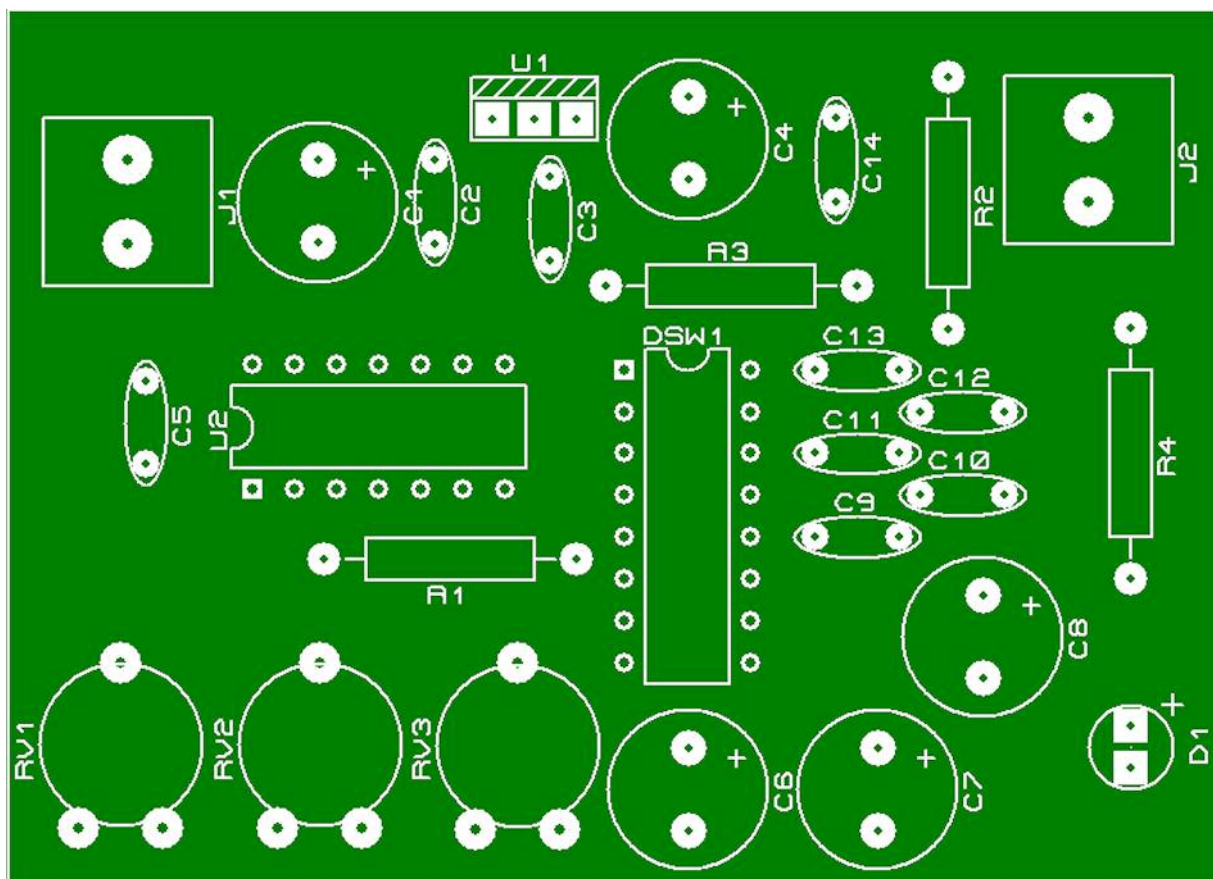



Figura 8: Disposizione dei componenti.

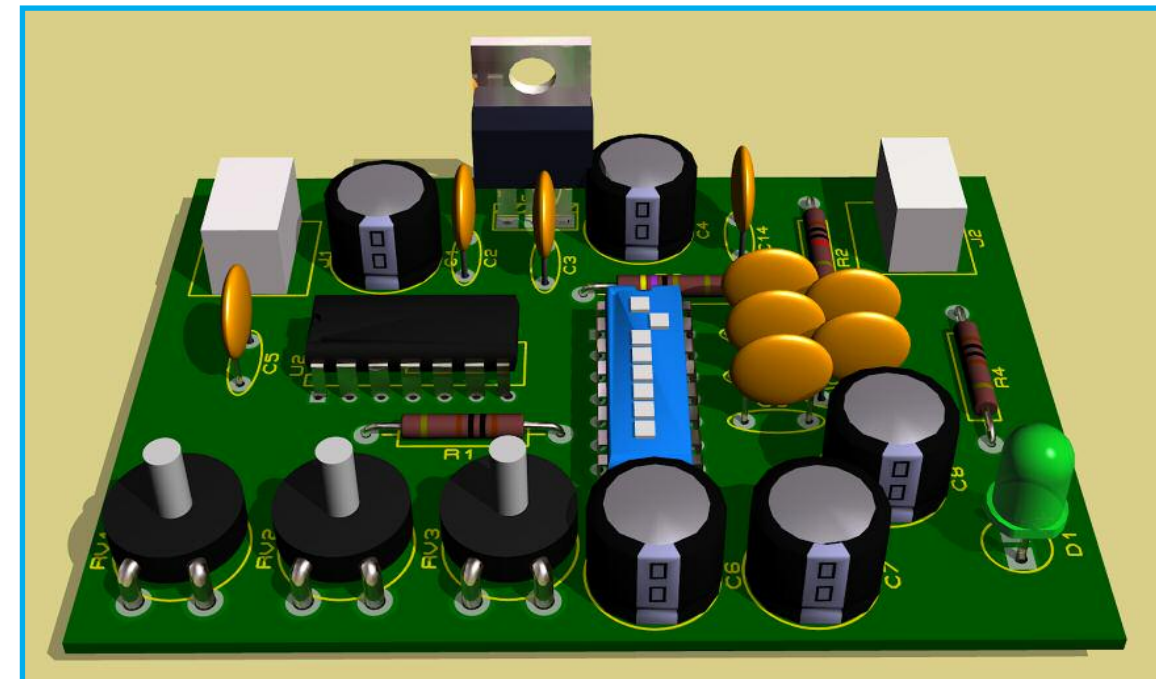



Figura 9: La scheda montata.

Come detto nell'introduzione, il generatore può avere tantissime applicazioni pratiche.

A grandi linee ecco a cosa esso può servire:

- A dare una base dei tempi digitale regolabile con continuità;
- A testare applicazioni a microcontrollore che abbiano ingressi digitali impulsivi;
- A far lampeggiare dispositivi luminosi a diverse frequenze;
- A provare applicazioni strobo;
- A generare una portante variabile (ricca di armoniche);
- A pilotare dispositivi di potenza;
- E molti, molti altri...

Il suo funzionamento può essere verificato in diversi modi:

- Collegando alle sue uscite un diodo Led (solo alle basse frequenze);
- Collegando un piccolo altoparlante (solo alle basse frequenze);
- Collegando un oscilloscopio;
- Collegando un frequenzimetro;
- Collegando un tester (solo alle basse frequenze);

L'integrato 74HC14

Il circuito integrato 74HC14 mette a disposizione sei buffer invertenti con trigger di Schmitt. In questo caso il momento di "indecisione" risulta molto ridotto e l'oscillazione avviene senza problemi. Rispetto al Cmos di tipo CD4016, esso risulta molto più veloce e permette di raggiungere velocità di commutazione molto elevate. E' caratterizzato da una dissipazione molto bassa e si usa molto per la realizzazione di generatori di impulsi, multivibratori astabili e monostabili. E' disponibile in contenitori di tipo DIP14, SO14, (T)SSOP14 e DHVQFN14. Per maggiori informazioni si consiglia di consultare il datasheet ufficiale.

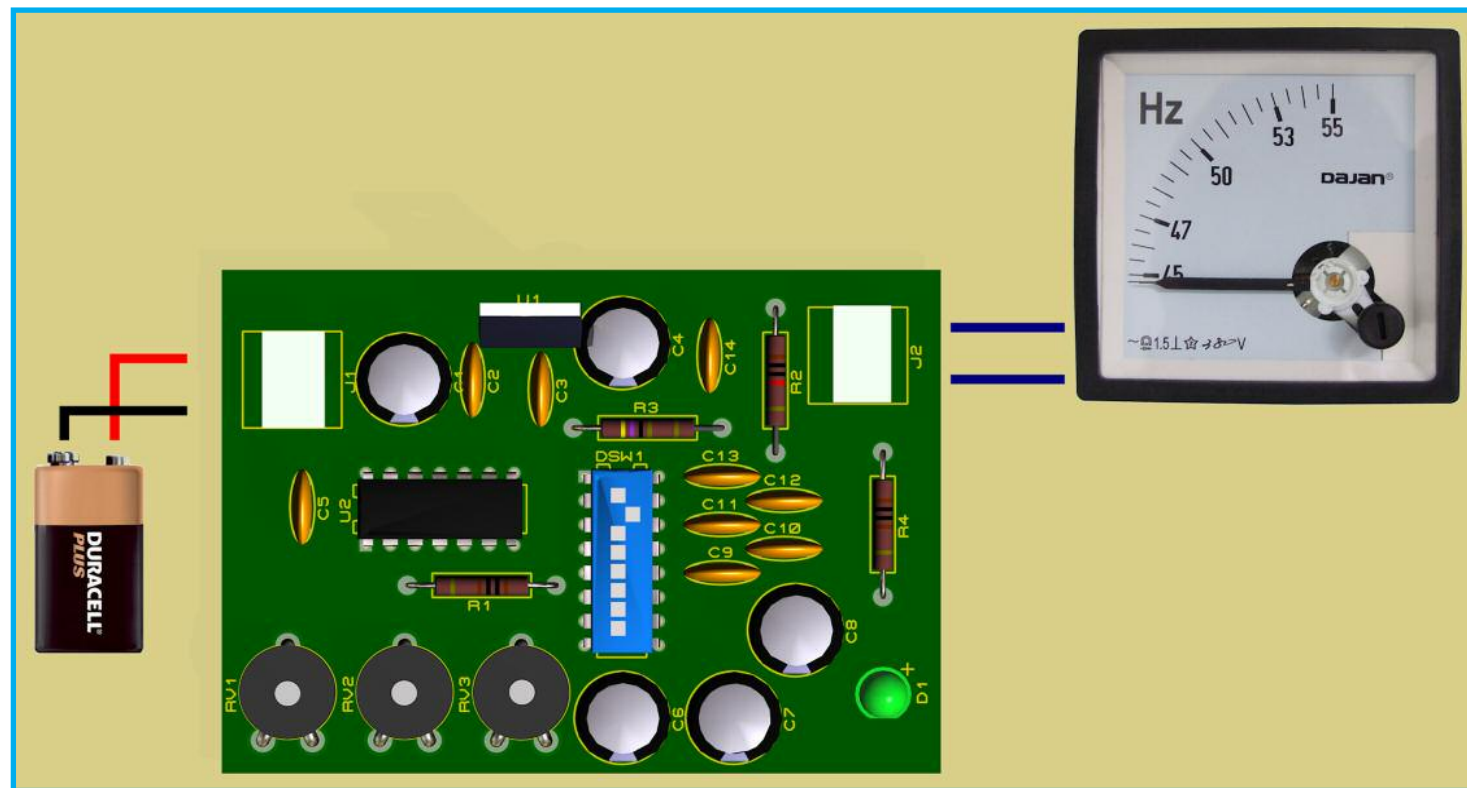



Figura 10: Collegamenti del generatore.

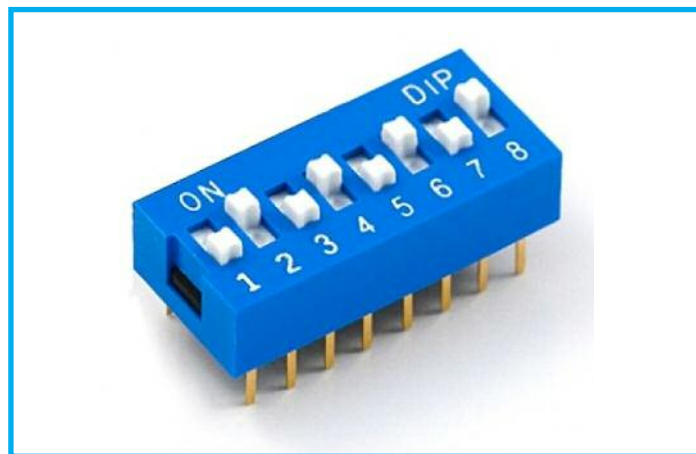



Figura 11: Il dip switch per la scelta della banda di frequenza.

Figura 12: Possibile soluzione di un contenitore.

- Avvicinandolo ad un ricevitore radio (solo alle alte frequenze);
- Eccetera.

CONCLUSIONI

La realizzazione del dispositivo proposto in queste pagine concorre ad arricchire il laboratorio dell'hobbista elettronico, specialmente per chi si occupa di elettronica digitale e di programmazione dei microcontrollori.
Si consiglia l'allestimento del circuito in un elegante contenitore, che possa conferire all'intero progetto anche un tocco di professionalità in più. In questo caso è opportuno sostituire i trimmer con altrettanti potenziometri, cablati con dei cavi schermati. Anche il diodo Led deve essere portato sul pannello, assieme ad un interruttore per attivare e disattivare l'alimentazione principale.
Si deve anche prevedere una boccola per l'uscita del segnale ad onda quadra sul pannello, possibilmente con un connettore BNC di tipo femmina e, in qualche modo, anche gli otto dip switch.

di VINCENZO SORCE

COMUNICAZIONE WIRELESS

CIRCUITO WI-COM-24

(parte seconda)

Il progetto Wi-Com-24, presentato nel numero scorso, consente il pilotaggio di relè via WiFi utilizzando, come interfaccia utente, un tablet PC Android. In questa seconda parte verrà illustrata la parte software del progetto

Dopo aver analizzato l'hardware del nostro progetto, nonché il funzionamento, ci addentriamo nella parte relativa alla programmazione. Essa è costituita da:

1) Un software, che consiste nella realizzazione di una GUI (Intefaccia Grafica Utente) tra il computer e la scheda comandata

2) Un firmware, che consiste nella memorizzazione, nel PIC della scheda comandata, di un programma di elaborazione dei dati ricevuti, di comando conseguente e di invio di controllo al PC

Adesso ci occuperemo del software utilizzando il linguaggio di programmazione PYTHON.

SOFTWARE CON PYTHON

In questa sede non è possibile, ovviamente, parlare in esteso di questo linguaggio di programmazione. Però illustreremo le caratteristiche principali ed i comandi essenziali che ci consentiranno di analizzare il programma elaborato.

Innanzitutto Python può definirsi un programma diverso da tutti gli altri. Infatti in questo linguaggio l'*indentazione*, che è un optional per gli altri programmi, è per esso fondamentale. L'*indentazione* consiste nell' impostare le varie righe di programma, come rientro, seguendo l'ordine di esecuzione. Ciò per consentire al programmatore di rileggerlo e controllarlo.

In pyhton le righe di programma saranno eseguite secondo l'indentazione. Un errore di indentazione causerà un errore di esecuzione.

E' un prezzo, però, che dà dei vantaggi enormi. Infatti non troverete tutte quelle parantesi, punteggiature etc. degli altri linguaggi di programmazione.

Un'altra caratteristica è quella della portabilità. Infatti è portabile in MS-DOS. Windows e OS/2. Con l'avvento degli smartphone e dei tablet sono nate delle prime applicazioni anche in *android.*

E' free, grazie all'opera del suo creatore, l'olandese Guido Van Rossum che nel febbraio del 1991 lo presentò al mondo.

Potremmo continuare a lungo ma non è l'obbiettivo di questo articolo.

1 START
2 IDENTIFICAZIONE UART
3 REALIZZAZIONE GUI
4 SCRITTURA E LETTURA MATRICE CARICHI
11 DIGITAZIONE NOME CARICO E PULSANTE OK
7 PULSANTI
8 DIGITAZIONE PULSANTE CARICO
12 LINK OFF->EXIT
9 COMANDO CARICO E REFRESH ON/OFF
10 REFRESH DI TUTTI I CARICHI
5 N
6 VIDEATA PROGRAMMA

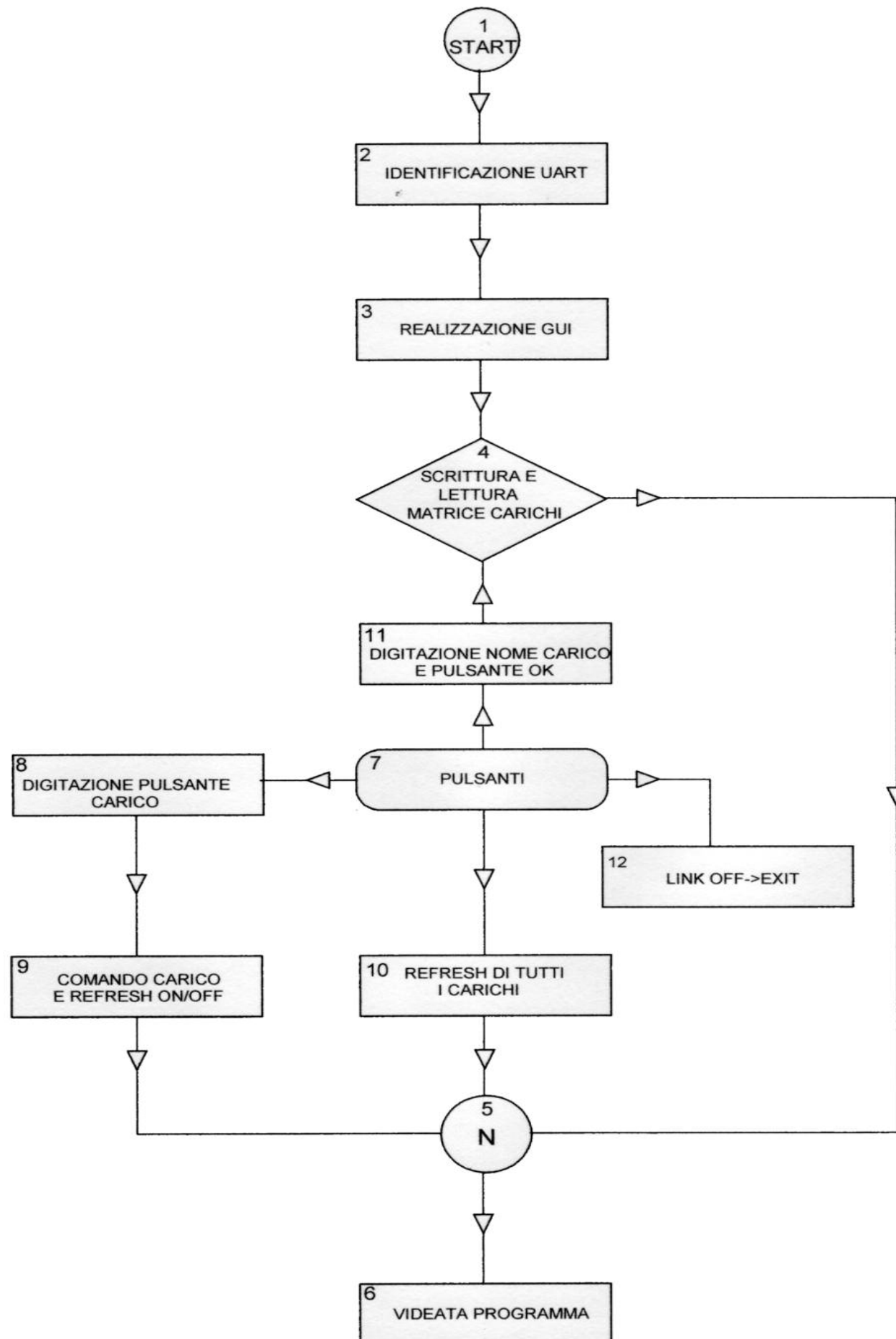



Figura 1: diagramma di flusso del programma.

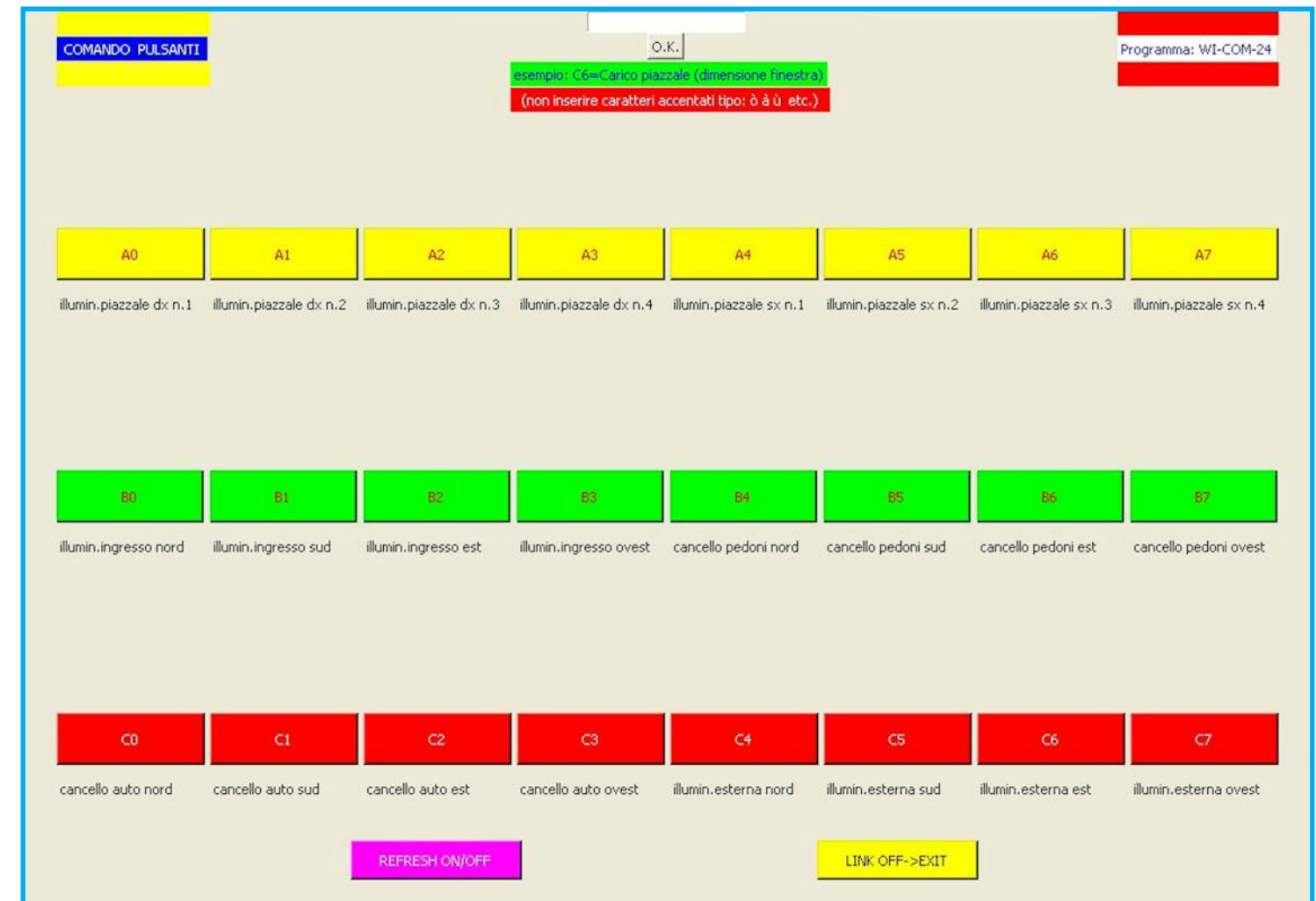



Figura 2: la schermata iniziale di avvio del software

Allora iniziamo l'esame del nostro programma.
Analizziamo la Figura 1 dove è rappresentata la flow chart (diagramma di flusso) del programma.
Quando il programma viene avviato ci troveremo in video quanto rilevabile dalla Figura 2.
Il flusso dei comandi si svolge nel seguente modo:

1) START
2) IDENTIFICAZIONE UART
3) REALIZZAZIONE GUI
4) SCRITTURA E LETTURA MATRICE CARICHI
5) nodo N di convergenza
6) VIDEATA PROGRAMMA
7) PULSANTI (nessun pulsante premuto)

Inizialmente premeremo il tasto *REFRESH ON/OFF* (punto 10). Allora la videata si aggiornerà in sintonia con lo stato dei carichi e si presenterà come in Figura 3.
Se premeremo un tasto di comando dei carichi si avrà la successione 7, 8, 9, 5, 6. Cioè il carico relativo commuterà e trasmetterà il nuovo stato ON/OFF che sarà visualizzato nella videata.
Se, invece, vogliamo variare il nome di un carico, digitiamo nella finestra superiore centrale il nuovo nome e premiamo il tasto OK. La successione sarà 7, 11, 4, 5, 6.
Se infine premiamo il tasto *LINK OFF->EXIT* , la UART sarà scollegata e potremo uscire dal programma cliccando sul *X* in

Figura 3: stato dei carichi collegati

alto a destra della videata.
Da quanto descritto appare evidente che la parte centrale di questo programma viene svolta con l'utilizzo dei pulsanti.. Ciò è dovuto alla scelta di utilizzare la libreria grafica *TKINTER* di Python. Infatti quest'ultima, oltre a disporre di comandi di scrittura, colori, screen etc., consente, con la definizione dei tasti, *def nometasto()*, di associare a ciascun tasto una serie di azioni decise dal programmatore.
Il programma in esame può apparire complesso analizzando il file sorgente. In realtà è molto ripetitivo dato che ripete gli stessi comandi associati a ciascun pulsante.
Naturalmente vi è uno stretto collegamento tra il software del PC ed il firmware della scheda comandata. Ci riserviamo di approfondire questo aspetto quando tratteremo il firmware di quest'ultima. Per semplificare la descrizione del programma che comanda l'unità master, lo abbiamo riportato nel listato 1 in modo semplificato.
Le prime righe sono relative all'associazione di moduli (librerie) che racchiudono determinate funzioni:

a) Tkinter: è la libreria grafica
b) Time: include le funzioni tempo
c) Serial: riguarda l'UART
d) Numpy: gestisce una matrice dati
e) Sys: è relativa ai file di sistema

Subito dopo viene definita, e aperta, una finestra, grazie al modulo Tkinter.

Poi viene aperta la porta seriale selezionata.
Si procede alla lettura iniziale dei carichi nello stato ON/OFF
Viene definito il tasto *prova* per l'inserimento del nome dei carichi. Se si ignora rimangono nella matrice carichi i dati memorizzati precedentemente.
Si noti che si fa riferimento solamente al tasto *A0.* Analogamente si procede per gli altri tasti. Il carico eventualmente inserito viene memorizzato con il comando *np.save("\wi-com-24\matrix",ar).*
Si noti la semplicità delle righe di programma e il rispetto rigoroso dell'*indentazione*.
Poi si procede alla definizione degli altri tasti. Anche in questo caso ci si riferisce ad A0.
Si noti, ad esempio, che la lettura matrice carichi ar=np.load("\wi-……..) è posta all'inizio della riga. Infatti viene letta all'avvio del programma.
Così come la definizione e la visualizzazione dei tasti.
Infine la finestra viene posta in *loop* (anello).
E' chiaro che si è voluto dare soltanto un'idea del programma. Il programma di installazione Python 2.7.3 e i moduli, insieme ai file sorgente, nonché il programma finale per versioni win XP e successive distribuzioni possono essere scaricati dal sito della rivista.

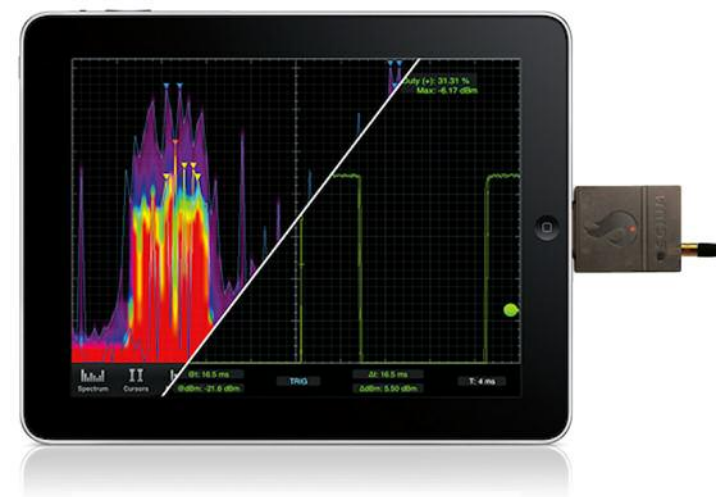
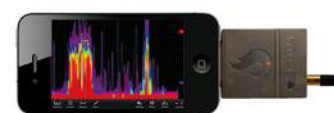
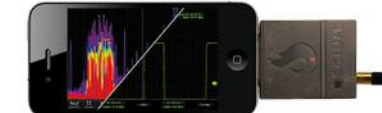
LISTATO 1

Download

```
######################################################################
#                                                                    #
#  PROGRAMMA WI-COM-24 - PYTHON 2.7                                  #
#  TX-RX  4800                                                       #
#                                                                    #
######################################################################
from Tkinter import *
#
import time
#
import serial
#
import numpy as np
#
import sys
#
finestra=Tk()            # apertura finestra
#
##############################  scansione porte ###############################
#
.....................................................................
.....................................................................
#
############################# fine scansione porte ###########################
#
ser = serial.Serial(PORTA,4800,8,"N",2,timeout=10)  # APERTURA PORTA
#
################ INSERIMENTO CARICHI E MEMORIZZAZIONE ###############
#
#
### inserimento carichi e memorizzazione
#
    if riga=="A" and colonna=="0":
        texa0=riman
        a0txt=Label(text="                              ")
        a0txt.pack()
        a0txt.place(x=35,y=220)
        a0txt=Label(text=texa0)
        a0txt.pack()
        a0txt.place(x=35,y=220)
```

```
        ar[0,0]=riman
        np.save("\wi-com-24\matrix",ar)
.......................................................................................
.................................................................
.......................................................................................
.................................................................
#
############## FINE INSERIMENTO CARICHI E MEMORIZZAZIONE #############
#
#
#################### FINESTRA SCRITTURA SUPERIORE ####################
#
.......................................................................................
.................................................................
.......................................................................................
.................................................................
#
######################## FINE SCRITTURA SUPERIORE #######################
#
#
############################ DEFINIZIONE TASTI  ###########################
#
############################# definizione tasto A0
#
def tastoa0():
    ser.write("a")
    time.sleep(0.1)
    ser.write("A")
    a0t=ser.read()
    #
    if a0t=="a":
        a0txt=Label(text="OFF",fg="blue",bg="white")
        a0txt.pack()
        a0txt.place(x=80,y=135)
    if a0t=="A":
        a0txt=Label(text="ON ",fg="red",bg="white")
        a0txt.pack()
        a0txt.place(x=80,y=135)
.......................................................................................
.................................................................
.......................................................................................
.................................................................

#
############################# fine definizione TASTI #########################
#
#
############################## ASSEGNAZIONE CARICHI ####################
#
#
ar=np.load("\wi-com-24\matrix.npy") # LETTURA MATRICE CARICHI
#

#
################# assegnazione carichi
#
a0txt=Label(text=ar[0,0])
a0txt.pack()
a0txt.place(x=35,y=220)
.......................................................................................
.................................................................
.......................................................................................
.................................................................
#
######################### FINE ASSEGNAZIONE CARICHI ####################
#
#
############################ ASSEGNAZIONE PULSANTI #####################
#
#
a0=Button(text="      A0      ",fg="red",bg="yellow",command=tastoa0,padx=30,pady=10)
a0.pack()
a0.place(x=35,y=170)
.......................................................................................
.................................................................
.......................................................................................
.................................................................
#
####################### FINE ASSEGNAZIONE PULSANTI #####################
#
finestra.mainloop()
#ser.close()
```

Prossimamente

Intervallometro fotografico

Per gli appassionati di fotografia e di elettronica ecco un interessante progetto con Arduino per realizzare scatti fotografici di sicuro impatto.

Orologio programmabile

Con un PIC ed il compilatore Mikropascal potrete realizzare questo orologio programmabile. Sarete anche liberi di personalizzarlo a vostro piacimento ede aggiungere ulteriori funzionalità.

Carico temporizzato

Una semplice ed utile realizzazione con un timer 555 che alla pressione di un pulsante attiva un carico e lo disattiva dopo alcuni secondi.

Raspberry Cyclette Pi

Immaginate cosa potreste fare se riusciste a collegare la cyclette di casa ad un Raspberry Pi…

Indagine sui lettori

Aiutaci
a conoscerti meglio!

Con il tuo aiuto riusciremo a offrirti una rivista sempre più in linea con le tue aspettative.

Compila online il questionario all'indirizzo
www.farelettronica.com/survey

Per ringraziarti **per il tuo tempo e la tua cortesia, ti invieremo** gratuitamente **un bellissimo** eBook **del valore di** 14,52 euro!

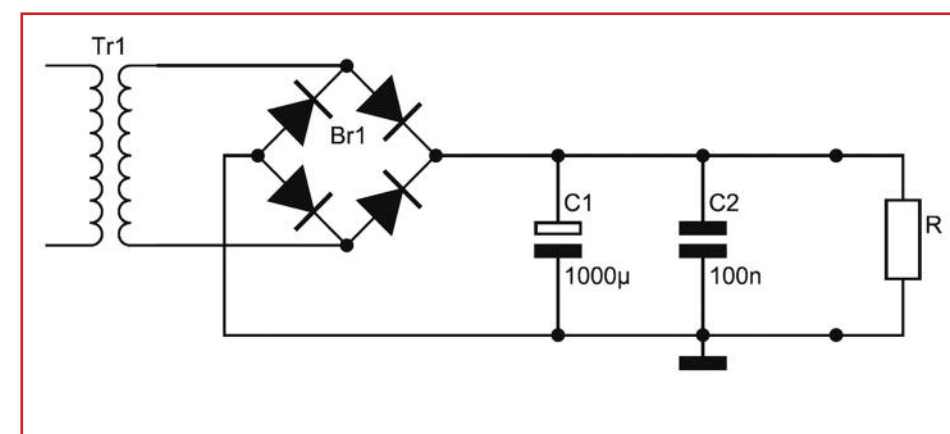
ElettroQuiz n. 337/338

FACILE

Per alimentare il carico R viene realizzato l'alimentatore non stabilizzato riportato in figura. L'alimentatore è a filtro LC ma nella progettazione è stato commesso un errore imperdonabile. Sapreste individuarlo?

IN PALIO:
Membership
Club Full

DIFFICILE

Determinare il valore della tensione di ripple sull'uscita dell'alimentatore non stabilizzato di figura se la corrente nel carico è Io=100mA.

IN PALIO:
Membership
Maker Full

INVIA LE TUE RISPOSTE

Le risposte ai quiz "facile" e "difficile" vanno inviate esclusivamente compilando il modulo su www.farelettronica.com/eq specificando la parola chiave "**marconi**".

Le risposte e i vincitori (previa autorizzazione) sono pubblicati alla pagina www.farelettronica.com/eq a partire dal 15 del mese successivo alla pubblicazione sulla rivista.

A tutti i partecipanti verrà assegnato un buono sconto del 10% (validità 3 mesi dalla data di assegnazione) utilizzabile per un prossimo acquisto su **www.ieshop.it**

di ANTONIO GIANNICO

CORSO MIKROPASCAL PER PIC
INTERRUPT E TIMER II

(parte nona)

Dopo aver fornito, nella precedente puntata, le basi teoriche sugli interrupt integrati nei controllori PIC, specie quelli della serie 16F, illustreremo in questa, con alcun esempi, l'impiego pratico degli interrupt on-change, dell'external interrupt RB0 e degli interrupt timer. Lo scopo è quello di imparare a utilizzare correttamente e in maniera efficace queste importantissime risorse nella progettazione delle applicazioni firmware

Dopo aver fornito, nella precedente puntata, le basi teoriche sugli interrupt integrati nei controllori PIC, specie quelli della serie 16F, illustreremo in questa alcune applicazioni pratiche di esempio riguardanti in particolare gli *interrupt on-change* e gli *interrupt timer*. Al fine di comprendere meglio l'importanza degli interrupt nell'affrontare e risolvere efficacemente determinate problematiche applicative dei controllori, mostreremo come gli esempi di codice MikroPascal della lezioni 7 possano essere riscritti utilizzando appunto gli interrupt. Gli esempi di codice mostrati in quella lezione e le nozioni fornite nella precedente lezione 8 sono pertanto da considerare a tutti gli effetti la base essenziale cui faremo riferimento nel corso dell'esposizione.
Tutti gli esempi applicativi presentati nelle precedenti puntate hanno come aspetti ricorrenti l'intercettazione di uno stato logico in ingresso o di una sua transizione da livello basso a livello alto o viceversa oppure temporizzazioni basate sull'impiego di ritardi implementati attraverso le funzioni di *built-in delay_ms* e *delay_us*. Quando si costruiscono applicazioni firmware un po' più complesse, specie algoritmi che consentono di gestire più funzionalità contemporaneamente, è spesso opportuno impiegare, nell'implementazione, timer e interrupt per i seguenti motivi:

- una temporizzazione basata su ritardi funziona correttamente fino a che la funzione di ritardo inserita all'interno del ciclo principale del programma viene ripercorsa regolarmente. Questa condizione viene meno quando all'interno dello stesso ciclo debbano essere gestite altre funzioni, per esempio quando conseguentemente alla pressione di un tasto il flusso di codice debba essere deviato verso una routine di interfaccia uomo macchina o debbano essere attivate operazioni e funzionalità che modificano momentaneamente la temporizzazione del ciclo principale rallentandone l'esecuzione in modo non esattamente prevedibile e fissa.
- l'intercettazione di uno stato che si presenta su un ingresso in maniera improvvisa e asincrona e che permane per un tempo molto ridotto non può essere gestito facilmente con un controllo a polling soprattutto se il ciclo principale di programma è complesso. Il polling comporterebbe infatti in questi casi il rischio di non intercettare in tempo l'evento specie se questo si estingue rapidamente. La gestone a polling dell'evento comporta inoltre e in generale una continua verifica dello stato del segnale e un conseguente eccessivo appesantimento dell'esecuzione.

Queste limitazioni possono essere agevolmente superate mediante l'impiego di interrupt tra cui i seguenti:

- quelli attivati dal cambiamento di stato di una porta (external interrupt e interrupt on-change);
- quelli attivati sull'overflow di un timer.

Si tratta di interrupt comunemente integrati in molti controllori PIC compreso il PIC16F877 utilizzato fino questo momento per i nostri esperimenti. Attraverso semplici esempi firmware applicativi, mostreremo come utilizzare queste risorse per ottenere le seguenti principali finalità:

- intercettazione a interrupt della variazione di stati logici presenti sulle porte di

I/O del controllore;
- gestione di temporizzazioni indipendenti dal flusso principale di programma.

```
(*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_1-Rev.0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma mostra il funzionamento di un interrupt su rising
   edge del pin RB0 utilizzato come ingresso.Il numero di transizioni in salita
   su RB0 viene contato e rappresentato su display a sette segmenti.I catodi
   comuni sono pilotati attraverso la porta B. L'assegnazione dei pin è fatta in
   maniera da rendere il codice compatibile con PICSIMLAB3. L'esempio è simile a
   quelli mostrati nella lez.8 con la differenza che l'incremento del conteggio
   è effettuato mediante interrupt su RB0 e non mediante controllo a polling
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *)
program Lezione9_ES_1;
var cnt : integer;            // variabile contatore
    stato_incremento : bit;   // se =1 allora il contatore cnt ha appena subito
    numero : word;            // un incremento
    digit: array[4] of word;  // array dei digit componenti il val. del contatore
    codice: array[4] of byte; // array dei codici per la rappresentazione su
                              // display a sette segmenti
function codifica(num: Word): Word;  // funzione di codifica del digit
begin
  case num of
    0 : result:= $3F;
    1 : result:= $06;
    2 : result:= $5B;
    3 : result:= $4F;
    4 : result:= $66;
    5 : result:= $6D;
    6 : result:= $7D;
    7 : result:= $07;
    8 : result:= $7F;
    9 : result:= $6F;
   end;
end;
//________________________________INTERRUPT__________________
procedure interrupt();
begin
 if(INTCON.INTF = 1 )then
  begin
    cnt:=cnt+1;         // Incremento della variable cnt
    stato_incremento:=1;
    ClearBit(INTCON, INTF);
  end;
  end;
//________________________________ MAIN _____________________
begin
cnt:=0;               // variabile contatore
TRISB := 0x01;        // Imposta RB0 come ingresso
TRISD := 0x00;        // Imposta PortD come uscita;
INTCON.GIE := 1;      // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;     // INTE: RB0/INT External Interrupt Enable bit
INTCON.INTF := 0;     // INTF: RB0/INT External Interrupt Flag bit
OPTION_REG.INTEDG:=1;// l'interrupt RB0/INT viene scatenato sul fronte di salita
                      // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                      // scatenato sul fronte di discesa
portB:= 0x00;         // inizializazione (spegnimento dei display a 7 segmenti)
stato_incremento:=1; // inizializzaione della variabile di stato
while(TRUE) do
          begin
              if (stato_incremento=1) then
                             begin
                             digit[3] := cnt div 1000 ;       //migliaia
                             digit[2] := (cnt div 100) mod 10;//centinaia
                             digit[1] := (cnt div 10) mod 10; //decine
                             digit [0]:= cnt mod 10;
                             codice[3]:= codifica(digit[3]);  //codice
                             codice[2]:= codifica(digit[2]);  //codice
                             codice[1]:= codifica(digit[1]);  //codice
                             codice[0]:= codifica(digit[0]);  //codice
                             stato_incremento:=0;
                             end;
              portD:= codice[3];
              portB:=0x80;        // selezione del display delle migliaia
              delay_ms(1);        // catodo comune pilotato da RB7
              portD:= codice[2];
              portB:=0x40;        // selezione del display delle centinaia
              delay_ms(1);        // catodo comune pilotato da RB6
              portD:= codice[1];
              portB:=0x20;        // selezione del display delle decine
              delay_ms(1);        // catodo comune pilotato da RB5
              portD := codice[0];
              portB:=0x10;        // selezione del display delle untà
              delay_ms(1);        // catodo comune pilotato da RB4
              if (cnt > 9999) then
                    cnt := 0;     // reinizio del conteggio
          end;
end. //______________________END MAIN_______________________
```


Figura 1: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES1 (impiego dell'external interrupt RB0/INT); si faccia riferimento alla Figura 2 per l'implementazione hardware dell'applicazione

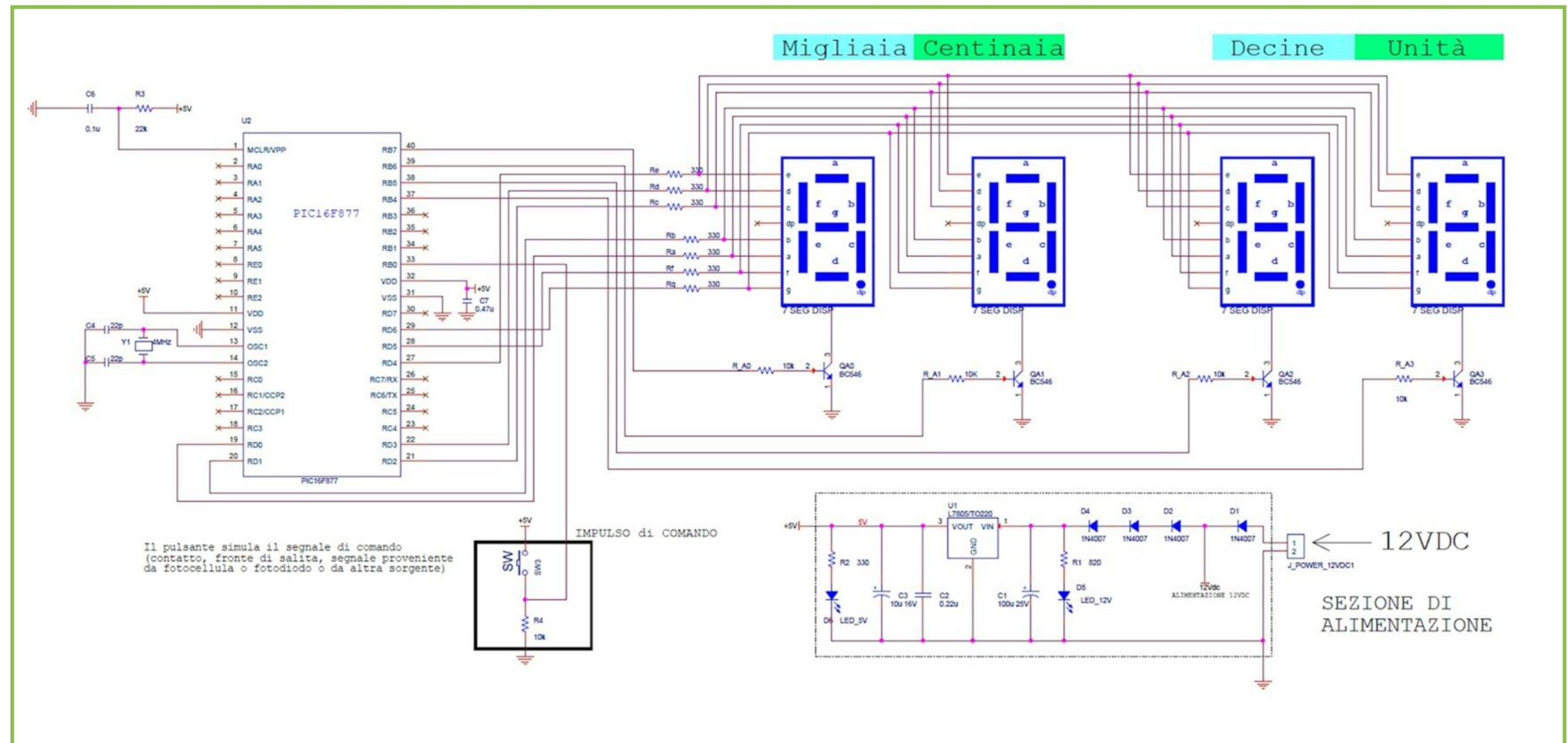



Figura 2: Schema circuitale implementativo dell'applicazione Lezione9_E1 e dell'applicazione Lezone9_ES2; si faccia riferimento ai listati di codice di Figura 1 e Figura 5 per gli aspetti firmware.

ESEMPI APPLICATIVI

Di seguito mostreremo sei esempi firmware applicativi scritti in linguaggio MikroPascal finalizzati a perseguire gli scopi illustrati nel precedente paragrafo introduttivo della presente puntata del corso MikroPascal PRO for PIC. Allo scopo non mancheremo di fare diretto riferimento alle precedenti due lezioni (numeri 7 e 8) e di riportare gli schemi dell'elettronica e i listati MikroPascal delle applicazioni.

Esempio 1-External Interrupt RB0/INT

Il programma di cui mostriamo in Figura 1 il listato MikroPascal e in Figura 2 l'implementazione circuitale, mostra il funzionamento dell'*external interrupt RB0/INT* impostato per essere scatenato su *rising edge* (fronte di salita).

Sul pin RB0 insiste un pulsante la cui pressione e rilascio genera una transizione di stato logico in salita seguita da una transizione in discesa.

Il numero di transizioni in salita su RB0 viene contato e rappresentato su una batteria di quattro display a sette segmenti (è pertanto possibile rappresentare un numero compreso tra 0 e 9999). I display sono pilotati secondo la tecnica del multiplexing trattata nella lezione n.7, i catodi comuni che attivano ciclicamente i singoli display a sette segmenti sono pilotati attraverso la porta B mentre i segmenti sono pilotati attraverso la porta D.

L'assegnazione dei pin è fatta in maniera da rendere il codice compatibile con PICSIMLAB3.

L'esempio è simile a quelli mostrati nella lezione n.7 con l'unica differenza che l'incremento del conteggio è effettuato intercettando la pressione del pulsante (che insiste sul pin RB0) sfruttando l'interrupt RB0/INT invece di fare affidamento sul controllo a polling.

Il codice può essere utilizzato, per esempio, per realizzare un contapezzi elettronico sostituendo al pulsante un circuito a fotocellula o a fotodiodo eccitato da un fascio luminoso che viene ostruito al passaggio degli oggetti da contare.

In questo esempio non è stato evitato del

```
//__________________________________________INTERRUPT_______
procedure interrupt();
begin
  if(INTCON.INTF = 1 ) then
   begin
    cnt:=cnt+1;          // Incremento della variable cnt
    stato_incremento:=1;
  //   ClearBit(INTCON, INTF);
   end;
   end;
```

Figura 3: e si modifica la procedura di interrupt dell'esempio di codice di Figura 1 (Lezione9_ES1), il conteggio non funziona correttamente in quanto l'interrupt flag resta sempre a valore 1 dopo la prima transizione di RB0 anche se sul pulsante che insiste su RB0 non viene effettuata alcuna azione

```
TRISB := 0x01;         // Imposta RB0 come ingresso
TRISD := 0x00;         // Imposta PortD come uscita;
INTCON.GIE := 1;       // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;      // INTE: RB0/INT External Interrupt Enable bit
INTCON.INTF := 0;      // INTF: RB0/INT External Interrupt Flag bit
OPTION_REG.INTEDG:=1;// l'interrupt RB0/INT viene scatenato sul fronte di salit
                       // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                       // scatenato sul fronte di discesa
```

Figura 4: Impostazione dei bit di configurazione degli interrupt (Lezione9_ES1, listato di Figura 1)

tutto l'impiego di delay a causa della necessità di gestire il multiplexing dei display a sette segmenti. Nell'esempio Lezione9_ES2 mostreremo come modificare ulteriormente il codice al fine di aggirare questa limitazione. A tale scopo introdurremo l'impiego di un timer per la temporizzazione del multiplexing.

Il controllore PIC utilizzato è il solito 16F877. Lo stato dell'external interrupt RB0/INT è definito dal Flag bit 1 del registro di interrupt INTCON (si faccia riferimento alla Tabella 1 della precedente lezione 8 per maggiori dettagli). Se tale bit assume valore 1 vuol dire che si è verificata la transizione in salita dello stato del pin RB0 utilizzato come ingresso, diversamente se tale bit rimane in stato zero vuol dire che non è avvenuta alcuna transizione. E' importante osservare come una volta scatenato l'interrupt e quindi "entrati" nella procedura di interrupt, non viene solo eseguito quanto previsto per la risoluzione stessa dell'interrupt (in questo caso l'incremento della variabile cnt) ma anche l'azzeramento dello stesso bit di interrupt INTF. Questa operazione deve avvenire infatti in firmware prima di uscire dalla procedura di interrupt in modo da rendere il processo pronto a ripetersi nel caso in cui l'interrupt venga scatenato una seconda volta. E' facile immaginare il malfunzionamento che si avrebbe se tale azzeramento non venisse effettuato o avvenisse erroneamente al di fuori della procedura di interrupt.

Modificando la routine di interrupt come mostrato in Figura 3 infatti, il conteggio non funziona correttamente in quanto l'interrupt flag resta sempre abilitato ad 1 dopo la prima transizione di RB0 anche se sul pulsante che insiste su RB0 non viene effettuata alcuna ulteriore azione.

Analizziamo in dettaglio alcune righe di codice relative alla gestione dell'interrupt. In particolare, osserviamo le righe di codice che vanno dalla numero 52 alla numero 55 (Figura 4). Invertiamo uno alla volta i valori assegnati rispettivamente ai bit GIE, INTE, INTF ed INTEDG e ogni volta ricompiliamo il codice e lo rieseguiamo al fine di verificare se l'applicazione funzioni ancora correttamente o meno.

L'istruzione INTCON.GIE:=1 assegna valore 1 al bit GIE del registro INTCON (bit 7 del registro). Questa operazione è necessaria in quanto il bit GIE (*Global Interrupt Enable bit*) deve sempre essere posto a valore 1 se si vuole effettuare l'abilitazione generale degli interrupt (GIE=0 corrisponde invece a disabilitazione generale degli interrupt). Commentando la riga di codice INTCON.GIE:=1 oppure sostituendola con INTCON.GIE:=0, ricompilando e rieseguendo il codice è possibile osservare come l'interrupt resti inattivo, di conseguenza il conteggio basato sulla transizione in salita di RB0 resta bloccato a valore zero.

All'interrupt su RB0 non è legato solo il bit INTF del registro di interrupt INTCON. Infatti, mentre INTF segnala che si è verificato l'interrupt su RB0, INTE (*RB0/INT External Interrupt Enable bit*) consente l'abilitazione dell'interrupt RB0/INT (INTE=1) o la disabilitazione dello stesso (INTE:=0). Se sostituiamo la riga di codice INTCON.INTE:=1 con INTCON.INTE:=0 pertanto, l'interrupt su RB0 rimane disabilitato e il flag INTF rimane inattivo. Stessa cosa accade se la riga di codice viene commentata.

L'istruzione INTCON.INTF:=0 è invece un'inizializzazione. Se si ponesse INTCON.INTF:=1 infatti, il codice inizierebbe la sua esecuzione con il flag a valore alto cosa che verrebbe interpretata come interrupt avvenuto su RB0 da cui scaturirebbe un'immediata partenza da valore 1 del contatore anche se non vi è stata ancora alcuna reale transizione sul pin RB0.

Qualche attenzione in più merita la riga di codice OPTION_REG.INTEDG:=1 in quanto trattasi del bit 6 del registro OPTION_REG di cui non abbiamo parlato nella precedente lezione 8. INTEDG sta infatti per *Interrupt Edge Select bit*. Se il suo valore viene posto ad 1 l'Interrupt si scatena sulla transizione in salita del segnale presente su RB0/INT; al contrario se è posto a 0 l'Interrupt si scatena sulla transizione in discesa. Se per esempio sul pin RB0 insiste un pulsante di pull-up allora la pressione del pulsante (supponendo di aver utilizzato l'impostazione OP-

Evento	Avvio programma	Primo overflow di TMR0 (RMR0IF=1)	Secondo overflow di TMR0 (TMR0IF=1)	Terzo overflow di TMR0 (RMR0IF=1)	Quarto overflow di TMR0 (RMR0IF=1)	...	
Clock Counter	0	1128	2152	3176	4200	...	
Time	0	141us	269 us	397 us	525 us	...	
DeltaClock	--	-	1024	1024	1024	...	...
DeltaT	-	-	128 us	128 us	128 us	...	...

Tabella 1-Evoluzione temporale del Timer0 (eventi di overflow valutati per il firmware dell'esempio Lezione9_ES2) analizzata attraverso l'impiego di PIC Simulator IDE.

TION_REG.INTEDG:=1) determinerà lo scatenarsi dell'interrupt. Al contrario se insiste su un pulsante di pull-down allora l'interrupt si scatenerà dopo la pressione ed il rilascio del pulsante.
Ovviamente, le cose si invertono se utilizziamo l'impostazione OPTION_REG.INTEDG := 0. Ovviamente, il tutto è rigorosamente vero a meno degli eventuali rimbalzi elettrici del contatto.
Il pulsante del resto è stato preso in considerazione in questo esempio solo perché rende il circuito facilmente implementabile ma in generale occorre pensare al segnale che insiste sul pin RB0 come proveniente dall'esterno da una generica sorgente compatibile con le specifiche elettriche del pin RB0 del controllore.

Esempio 2-External Interrupt RB0/INT e Timer0

In questo esempio vogliamo mostrare come sia possibile utilizzare un timer. In particolare, utilizzeremo il Timer0 per la gestione in multiplexing dei quattro display a sette segmenti dell'esempio "Lezione9_ES1" unitamente all'external interrupt RB0/INT. Così facendo avremo modo di mostre come sia possibile utilizzare e gestire più interrupt nella stessa applicazione.
Il timer consente in questo esempio di eliminare l'impiego dei delay nella gestione in multiplexing dei display a sette segmenti mentre l'interrupt su RB0 viene utilizzato per implementare il solito algoritmo contapezzi già mostrato con Lezione9_ES1. In questo caso è necessario implementare la procedura di interrupt in maniera tale da distinguere in firmware, ogni volta che viene scatenato un interrupt, quale degli interrupt possibili sia stato effettivamente scatenato al fine di eseguire le corrispondenti operazioni di risoluzione e solo quelle. L'implementazione hardware dell'applicazione è la stessa mostrata in Figura 2 mentre quella firmware è mostrata in Figura 5. In particolare, è utile confrontare il listato con quello dell'esempio precedente al fine di osservare due sostanziali differenze:

- il modo in cui vengono discriminati i due interrupt all'interno di una comune procedura di interrupt (si noti l'istruzione di controllo if..then.. else…);
- l'impiego e l'impostazione del Timer0.

L'interrupt legato al timer viene scatenato ogni volta che il timer (timer0) va in overflow. Per questo motivo è possibile "ag-

```
{*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_2-Rev.0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma mostra il funzionamento di un interrupt su rising
   edge del pin RB0 utilizzzato come ingresso.Il numero di transizioni in salita
   su RB0 viene contato e rappresentato su display a sette segmenti.I catodi
   comuni sono pilotati attraverso la porta B. L'assegnazione dei pin è fatta in
   maniera da rendere il codice compatibile con PICSIMLAB3. L'esempio è simile a
   Lezione9_ES_2. In questo caso il multiplexing dei display a 7 segmenti
   è gestito temporalmente attraverso l'impiego del Timer0.
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *)
program Lezione9_ES_2;
var cnt, t7seg: integer;      // cnt: variabile contatore
    numero : word;            // t7seg: scandisce gli intervalli di multiplexing
    digit: array[4] of word; // delle cifre componenti il contatore
    codice: array[4] of byte;

function codifica(num: Word): Word;  // funzione di codifica del digit
begin
  case num of
    0 : result:= $3F;
    1 : result:= $06;
    2 : result:= $5B;
    3 : result:= $4F;
    4 : result:= $66;
    5 : result:= $6D;
    6 : result:= $7D;
    7 : result:= $07;
    8 : result:= $7F;
    9 : result:= $6F;
  end;
end;
//_____________________________________INTERRUPT___________________
procedure interrupt();
begin
if TestBit(INTCON, TMR0IF) = 1 then
        begin
          t7seg := t7seg + 1;
          ClearBit(INTCON, TMR0IF);        // azzeramento del Flaf del Timer0
        end
        else
        if(INTCON.INTF = 1 ) then
                begin
                cnt := cnt + 1;          // incremento del contatore
                ClearBit(INTCON, INTF);  // Azzeramento del flag INTF
                end;
end;
//_____________________________________MAIN PROGRAM___________________
begin
cnt:=0;                 // inizializzaizone variabile contatore
t7seg:=0;               // iizializzazione del contat. che scandisce il multiplexing
TRISB := 0x01;          // Imposta RB0 come ingresso
TRISD := 0x00;          // Imposta PortD come uscita;
INTCON.GIE := 1;        // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;       // INTE: RB0/INT External Interrupt Enable bit
INTCON.PEIE := 0;
INTCON.INTF := 0;       // INTF: RB0/INT External Interrupt Flag bit
OPTION_REG.INTEDG := 1; // interrupt RB0/INT è scatenato sul fronte di salita
                        // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                        // scatenato sul fronte di discesa
OPTION_REG.T0CS := 0;   // Fosc/4 come sorgente per il timer
INTCON.TMR0IE := 1;     // TMR0 Interrupt enable
OPTION_REG.PSA:=1;      // nessun prescaler
TMR0 := 0;              // inizializzazione del TMR0
while(TRUE) do
          begin
           digit[3] := cnt div 1000 ;        // cifra delle migliaia
           digit[2] := (cnt div 100) mod 10;// cifre delle centinaia
           digit[1] := (cnt div 10) mod 10; // cifre delle decine
           digit[0] :=  cnt mod 10;
           codice[3]:= codifica(digit[3]);  // codice cifra migliaia
           codice[2]:= codifica(digit[2]);  // codice cifra centnaia
           codice[1]:= codifica(digit[1]);  // codice cifra decine
           codice[0]:= codifica(digit[0]);  // codice cifra unità
           while (t7seg<25)AND(t7seg>=0) do
                  begin
                   portD:= codice[3]; // digit delle migliaia
                   portB:=0x80;  // catodo comune pilotato da RB7
                  end;
           while (t7seg<50)AND(t7seg>=25) do
                  begin
                   portD:= codice[2]; // digit delle centinaia
                   portB:=0x40; // catodo comune pilotato da RB6
                  end;
           while (t7seg<75)AND(t7seg>50) do
                  begin
                   portD:= codice[1]; // digit delle decine
                   portB:=0x20; // catodo comune pilotato da RB5
                  end;
           while (t7seg<100)AND(t7seg>75) do
                  begin
                  portD := codice[0]; // digit delle unità
                  portB:=0x10;  // catodo comune pilotato da RB4
                  end;
           if (cnt > 9999) then
                    cnt := 0;           // reinizio del conteggio
           if (t7seg=100) then
                    t7seg := 0;         // reinizio ciclo di multiplexing
        end;
end. //                                  END MAIN
```

Device — Name: P16F877
MCU Clock — Frequency: 8.000000 MHz

Figura 5: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES2 (impiego dell'external interrupt RB0/INT e dell'interrupt Timer0); si faccia riferimento alla Figura 2 per l'implementazione hardware dell'applicazione

Figura 6: Schema logico-funzionale del Timer0. Lo schema è utile al programmatore al fine di eseguire le corrette impostazioni dei bit di configurazione (T0SE, T0CS, PSA, PS2:PS0)

ganciare" a questo evento (overflow del Timer0) l'incremento di un contatore che realizza in sostanza la temporizzazione.
E' opportuno fare alcune considerazioni utili sul tempo di overflow del timer0. Quando si deve implementare correttamente una o più temporizzazioni all'interno dell'applicazione è infatti necessario mettere correttamente in relazione il tempo di overflow del timer con il numero di overflow necessari per coprire un certo intervallo temporale. Questi due parametri sono a loro volta legati al clock con il quale il timer viene pilotato, clock che nella maggior parte dei casi coincide, a meno di un fattore di divisione, con il clock di sistema cioè del controllore. Con riferimento all'esempio Lezione9_ES2 (Figura 5-si notino i bit di configurazione del timer), se si analizza l'evoluzione temporale del processo firmware con l'ausilio di un simulatore come PIC Simulator IDE, è possibile osservare che questo va in overflow (flag TMR0IF) a intervalli temporali regolari che misurati in microsecondi e in termini di fronti di clock di sistema risultano pari ai valori riportati in Tabella 1.
Come si può osservare, il Timer0 va in overflow a cadenza regolare con periodo pari a 1024 colpi di clock corrispondenti a 128 us. Questa regolarità non è rispettata solo all'avvio del processo quando devono essere eseguite le istruzioni di inizializzazione del Timer, degli interrupt e delle variabili. La cadenza regolare osservata è facilmente spiegabile. Infatti, con riferimento allo schema logico funzionale del Timer0 (Figura 6) e alle righe di codice che definiscono l'impostazione del timer stesso (listato di Figura 5), è possibile osservare che:

- TOCS=0 (OPTION_REG.T0CS:=0;) per cui la sorgente di clock per il timer è Fosc/4 dove Fosc è la frequenza del clock esterno cioè di sistema imposto al controllore dalla sezione di oscillazione e quindi dal quarzo (8 MHz);
- INTCON.TMR0IE:=1; il timer0 è quindi abilitato;
- OPTION_REG.PSA:=1; non viene quindi imposto alcun prescaler al clock stesso prima che questo solleciti il timer;
- TMR0:= 0; il timer parte da valore zero quindi senza alcun valore precaricato.

Sulla base di questi dati è possibile affermare che il tempo di overflow inteso come il tempo necessario affinché il timer0 vada in overflow è pari a:

$$T_{of} = 256xT_{inc} = 256xT_{clock}x4 = 256x(1/F_{osc})x4 = 256x(1/8MHz)x4 = 128 \text{ us.}$$

dove T_{inc} è il tempo necessario ad un incremento del registro Timer0, T_{of} è il tempo di overflow dello stesso registro o meglio il tempo che intercorre tra due overflow successivi, T_{clock} è il tempo di clock di sistema pari all'inverso della frequenza di clock (1/8 MHz), il fattore 4 deriva dal fatto che il timer è sollecitato da un clock diviso in frequenza per 4 (TOCS=0) e 256 è dovuto al fatto che il timer0 è un timer ad 8 bit per cui va in overflow passando per 256 stati.
In conclusione, intercorrono 128 us (microsecondi) tra un overflow e l'altro del Timer0. Evidentemente, essendo il clock del Timer (Fosc/4) quattro volte più lento di Fosc, essendoci cioè un incremento ogni 4 colpi di clock, tra un overflow e l'altro intercorrono 4x256=1024 fronti di clock. I dati ricavati attraverso il simulatore e riportati in Tabella 1 risultano pertanto perfettamente spiegati.
Qualche ulteriore osservazione e verifica didatticamente interessante può essere fatta circa il tempo di multiplexing dei display a sette segmenti la cui gestione in questo esempio è resa possibile dalle temporizzazioni che poggiano sul timer0. Se si utilizza il solito simulatore PIC Simulator IDE è possibile verificare che l'attivazione dei singoli display a sette segmenti avviene, nell'ordine, secondo la sequenza temporale riportata in Tabella 2. Le distanze temporali tra gli eventi, come al solito sono espresse sia in termini di impulsi di clock di sistema (counter clock) che in termini di tempo assoluto a partire dall'avvio

Evento	Avvio programma	Attivazione digit3 migliaia	Attivazione digit3 centinaia	Attivazione digit3 decine	Attivazione digit3 unità	...
Clock Counter	0	1556	26160	53856	79404	...
Time	0	194,5us	3270 us	6732 us	9925 us	...
DeltaClock			24604	27696	25548	
DeltaT			3075,5 us	3462 us	3193 us	

Tabella 2-Evoluzione della temporizzazione del multiplexing dei display a sette segmenti valutata per il firmware dell'esempio 2 (Lezione9_ES2) attraverso l'impiego di PIC Simulator IDE.

Figura 7: Implementazione hardware dell'applicazione degli esempi firmware Lezione9_ES3 (Figura 8), Lezione9_ES4 (Figura 9) e Lezione9_ES6 (Figura 12). In questo ultimo listato di esempio è possibile aggiungere sul pin RD2 del controllore un led la cui accensione indica il perdurare del tempo di antibouncing. Ricompilando il codice con diversi valori di tempo di antibouncing è possibile studiare l'efficacia della tecnica di antibouncing.

dell'esecuzione del firmware (espresso in microsecondi).
Anche questi dati, sebbene rilevati con un minimo di approssimazione sono facilmente spiegabili ed interpretabili. Infatti, secondo il listato di codice, il tempo per il quale è mantenuto acceso un display è tale da coprire 25 overflow del Timer0.
Poiché un overflow si verifica solo dopo il passaggio per 256 stati di Timer0 (timer a 8 bit), essendo il segnale che sollecita il Timer di frequenza pari a Fosc/4 (vedi Figura 5 e Figura 6 e in particolare l'impostazione dei bit PSA e TOCS del timer nel listato) e la frequenza di clock pari ad 8MHz, il tempo di overflow del timer è pari a:

$$T_{of}=(1/8MHz)x4x256=128us$$

di conseguenza, il tempo di multiplexing inteso come tempo per il quale perdura l'accensione del singolo display a sette segmenti è pari a:

$$T_{mux}=25x128us=3,2 \text{ ms circa}$$

In effetti, il tempo misurato medio per cui resta acceso ciascun display (vedi Tabella 2) è pari a circa 3200us=3,2 ms. Tale tempo deve inoltre corrispondere a un numero di impulsi di clock pari a:

256 stati del Timer x 25 overflow del Timer x 4 imp. di clock = 25x1024 imp. di clock = 25600 imp. di clock

Tale valore corrisponde effettivamente al DeltaClock misurato e riportato nella stessa Tabella 2.
I calcoli appena fatti non vanno ovviamente presi come un puro esercizio matematico ma piuttosto come un procedimento teso a comprendere praticamente gli effetti delle impostazioni dei bit di configurazione dei timer sulle temporizzazioni.

Esempio 3 e Esempio 4-External interrupt RB0/INT e RB On-Change

Con questi due esempi intendiamo mostrare l'impostazione e l'impiego pratico degli interrupt *RB ON-Change* che per il PIC16F877 corrispondono ai bit più significativi della porta B (RB4:RB7).
L'applicazione è dettagliata in Figura 7 per quanto riguarda gli aspetti hardware e in Figura 8 per quelli firmware (codice MikroPascal).

```
{*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_3-Rev. 0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma mostra, con un semplice esempio, il funzionamento
   dell'External interrupt RB0/INT e degli Interrupt RB (4..7) ON-Change.
   Alla pressione di uno dei pulanti presenti sui pin RB4,RB5,RB6 e RB7 gestiti
   come interrupt on-change, corrispnde l'attivazione rispettiamente delle
   uscite RC1,RC2,RC3 ed RC4. La pressione del pulsante posto su RB0 determina
   la disattivazione simultanea di tutte le uscite.
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *}
program Lezione9_ES_3;
//___________________ // Procedura di Interrupt___________________
procedure interrupt();
begin
if(INTCON.INTF =1) then // Gestione External RB0/INT_______________
               begin
                 portc:=0x00;
                 ClearBit(INTCON, INTF);
               end
               else
if(INTCON.RBIF =1) then // Gestione RB On-Change ____
               begin
                 if portB.4=1 then portc.4:=1;
                 if portB.5=1 then portc.5:=1;
                 if portB.6=1 then portc.6:=1;
                 if portB.7=1 then portc.7:=1;
                 ClearBit(INTCON, RBIF);
               end;
end;
//_______________________________________MAIN PROGRAM_______________________
begin
TRISB := 0xFF;      // Imposta portB come ingresso
TRISC := 0x00;      // Imposta portC come uscita
portC := 0;         // Azzera uscita porta C
INTCON.GIE := 1;    // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;   // INTE: RB0/INT External Interrupt Enable bit
INTCON.INTF := 0;   // INTF: RB0/INT External Interrupt Flag bit
INTCON.PEIE := 0;   // PEIE: Peripheral Interrupt Enable bit
OPTION_REG.INTEDG:=1; //l'interrupt RB0/INT viene scatenato sul fronte di salita
                      // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                      // scatenato sul fronte di discesa
INTCON.RBIE := 1;     // RBIE: RB(4..7) Port Change Interrupt Enable bit
     while (true) do
           begin
       //    cilco vuoto
           end;
end.
//______________________________________END PROGRAM_______________________
```


Figura 8: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES3 (impiego dell'interrupt RB0/INT e degli interrupt RB ON-Change); si faccia riferimento alla Figura 7 per l'implementazione hardware dell'applicazione.

```
{*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_4-Rev. 0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma è equivalente Lezione9_ES3 ma è scritto
   in maniera formalmente più corretta. Mostra il funzionamento dell'External
   interrupt RB0/INT e degli Interrupt RB (4..7) ON-Change. Alla pressione di
   uno dei pulanti presenti sui pin RB4,RB5,RB6 e RB7 gestiti come interrupt
   on-change, corrispnde l'attivazione rispettiamente delle uscite RC1,RC2,RC3
   ed RC4. La pressione del pulsante posto su RB0 determina la disattivazione
   simultanea di tutte le uscite.
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *}
program Lezione9_ES_4;
var stato_porta:byte;
//___________________ // Procedura di Interrupt___________________
procedure interrupt();
begin
if(INTCON.INTF =1) then // Gestione External RB0/INT___
 begin
    stato_porta:=0;
    ClearBit(INTCON, INTF);
    end
 else
 if(INTCON.RBIF =1) then// Gestione RB On-Change ____
 begin
    stato_porta:=(stato_porta)OR(portB);
    ClearBit(INTCON, RBIF);
    end;
end;
//_______________________________________MAIN PROGRAM_______________________
begin
TRISB := 0xFF;      // Imposta RB come ingresso
TRISC := 0x00;      // Imposta RC come uscita
portC := 0;         // inizializza i valori della porta di uscita
stato_porta:=0;     // inizializza lo stato della porta
TRISB := 0xFF;      // Imposta portB come ingresso
TRISC := 0x00;      // Imposta portC come uscita
portC := 0;         // Azzera uscita portC
INTCON.GIE := 1;    // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;   // INTE: RB0/INT External Interrupt Enable bit
INTCON.INTF := 0;   // INTF: RB0/INT External Interrupt Flag bit
INTCON.PEIE := 0;   // PEIE: Peripheral Interrupt Enable bit
OPTION_REG.INTEDG:=1;//l'interrupt RB0/INT viene scatenato sul fronte di salita
                     // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                     // scatenato sul fronte di discesa
INTCON.RBIE := 1;    // RBIE: RB(4..7) Port Change Interrupt Enable bit
     while (true) do
                 begin
                   if stato_porta=0 then
                   portC:=(portC)AND(0x0F)
                                 else
                                 portC:=(portC)OR(stato_porta);
                 end;
end. //_______________________________________END PROGRAM_______________
```


Figura 9: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES4 (miglioramento dell'esempio Lezione9_ES3); si faccia riferimento alla Figura 7 per l'implementazione hardware dell'applicazione

Figura 10: Schema circuitale implementativo dell'applicazione Lezione9_ES5; si faccia riferimento al listato di codice di Figura 11 per gli aspetti firmware.

Alla pressione di uno dei pulsanti presenti sui pin RB4,RB5,RB6 e RB7 gestiti come interrupt on-change, corrisponde l'immediata attivazione rispettivamente delle uscite RC1,RC2,RC3 ed RC4. La pressione del pulsante posto su RB0 determina la disattivazione simultanea e istantanea di tutte le uscite.

Normalmente la procedura di interrupt dovrebbe agire in maniera immediata e ritornare il controllo al flusso principale di programma. In effetti, se non si segue questo modo di strutturare il codice, si finisce per appesantire la procedura di interrupt alleggerendo in maniera inutile il ciclo principale di programma come è accaduto nel presente esempio di Figura 8 in cui il ciclo "*while.. do*" principale è praticamente vuoto. Inoltre questo non solo appesantisce la procedura di interrupt ma rende l'intero codice meno elegante. E' allora opportuno modificare il codice portando tutte le istruzioni possibili al di fuori della procedura di interrupt. Il codice di Figura 8 è stato infatti volutamente scritto allo scopo come probabilmente lo scriverebbe una persona alle prime armi con la programmazione firmware dei controllori. Anche il codice del flusso principale di programma è però bene che sia il più compatto ed efficiente possibile. Il codice può allora essere modificato come riportato nell'esempio Lezione9_ES4 (Figura 9). Questo è quindi da preferire a quello di Figura 8 sebbene i due

```
{*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_5-Rev.0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma mostra il funzionamento di un interrupt Timer0
   utilizzato come contatore. Il segnale di conteggio è il fronte presente su
   RA4. Tale segnale si sostituisce al clock interno del controllore. Allo scopo
   è necessario impostare opportunamente i bit del registro OTION_REG
   (OPTION_REG.TOCS:=1; OPTION_REG.TOSE:=0).Per semplicità si considera un
   contatore ad un solo digit. Il valore è rappresentato su display a 7 segmenti
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *}
program Lezione9_ES_5;

function codifica(num: Word): Word;  // funzione di codifica del digit
begin
  case num of
     0 : result:= $3F;
     1 : result:= $06;
     2 : result:= $5B;
     3 : result:= $4F;
     4 : result:= $66;
     5 : result:= $6D;
     6 : result:= $7D;
     7 : result:= $07;
     8 : result:= $7F;
     9 : result:= $6F;
   end;
end;
//________________________________________INTERRUPT________
procedure interrupt();
begin
 if TestBit(INTCON, TMR0IF) = 1 then
        begin
         ClearBit(INTCON, TMR0IF);
        end;
end;

begin
TRISA:=0xFF;         // la porta A è impostata come ingresso
TRISB := 0x01;       // Imposta RB0 come ingresso
TRISD := 0x00;       // Imposta PortD come uscita;
INTCON.GIE := 1;     // GIE: Global Interrupt Enable bit
INTCON.TMR0IE := 1;  // TMR0 Interrupt Enable
OPTION_REG.TOCS:=1;  // Fosc/4 come sorgente per il timer
OPTION_REG.TOSE:=0;  // Timer0 Source Edge (su fronte di salita)
TMR0 := 0;           //  inizializzazione del TIMER0
while(TRUE) do
            begin
                portD:= codifica(TMR0);
                portB:=0x10;
                if (TMR0 > 9) then
                           TMR0:= 0; // reinizio del conteggio
            end;
end. //____________________________________END MAIN________
```

Device
Name: P16F877
MCU Clock
Frequency: 8.000000 MHz

Figura 11: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES5; si faccia riferimento alla Figura 10 per l'implementazione hardware dell'applicazione. Il Timer0 viene utilizzato come contatore degli impulsi che si presentano sul pin RA4.

presentino all'esterno lo stesso identico comportamento.

Completiamo la descrizione di questi due esempi notando (Figura 8 e Figura 9):

- l'impostazione della porta B come ingresso e della porta C come uscita;
- l'abilitazione del global interrupt (INTCON.GIE=1;);
- l'abilitazione dell'external interrupt RB0/INT (INTCON.INTE:= 1);
- l'impostazione del fronte del segnale (INTEDG=interrupt edge) su RB0 su cui scatenare l'interrupt RB0/INT (OPTION_REG.INTEDG:=1); l'impostazione effettuata abilita l'interrupt ad essere scatenato sul fronte di salita. Invertendo il valore dell'assegnazione si avrebbe l'attivazione dell'interrupt sul fronte di discesa;
- l'abilitazione dell'interrupt RB ON-Change (RBIE=1);
- il riconoscimento dell'evento di interrupt ON-Change (flag RBIF);
- l'impiego di operazioni bitwise che consentono di scrivere istruzioni estremamente compatte.

```
{*
 * Corso: MikroPascal PRO for PIC-Fare Elettronica-Lezione n.: 9
 * Nome progetto: Lezione9_ES_4-Rev. 0: 2013-04-16; Autore: A. Giannico
 * Descrizione: Il programma mostra l'implementazione firmware di un filtro
   temporale antibouncing basato sul timer TMR0. Il filtro è applicato agli
   ingressi RB0/INT ed RB4...RB7 gestiti come interrupt on-change.
   L'esempio è la naturale evoluzione di Lezione9_ES4 cui è aggiunto
   l'antibouncing.
 * MCU:PIC18F877; Sch. di sviluppo:Easypic3; Oscillatore: HS 8.0000 MHz
 * SW: mikroPascal PRO for PIC  ver. 5.61
   http://www.mikroe.com/eng/products/view/10/mikropascal-pro-for-pic/
 *}
program Lezione9_ES_6;
var stato_porta, antirimbalzo :byte;
var t_antirimbalzo: integer;
Const T_filtro: integer=400;
//_____________________ // Procedura di Interrupt_____________________
procedure interrupt();
begin
 if TestBit(INTCON, TMR0IF) = 1 then
        begin
          if (antirimbalzo=1) then
                              if (t_antirimbalzo<T_filtro) then
                                     t_antirimbalzo := t_antirimbalzo+1
                                     else
                                       begin
                                        antirimbalzo:=0;
                                        t_antirimbalzo:=0;
                                        portD.2:=0;
                                       end;
          ClearBit(INTCON, TMR0IF);
        end
else
if(INTCON.INTF =1) then // Gestione External RB0/INT_____________________
                  begin
                    if (antirimbalzo=0) then
                                        begin
                                         stato_porta:=0;
                                         antirimbalzo:=1;
                                         portD.2:=1;
                                        end;
                    ClearBit(INTCON, INTF);
                  end
                  else
 if(INTCON.RBIF =1) then// Gestione RB On-Change _____________________
                  begin
                    if (antirimbalzo=0) AND (portB<>0) then
                                        begin
                                          stato_porta:=(stato_porta)OR(portB);
                                          antirimbalzo:=1;
                                          portD.2:=1;
                                        end;
                    ClearBit(INTCON, RBIF);
                  end;
end;
//_____________________MAIN PROGRAM_____________________
begin
antirimbalzo:=0;
t_antirimbalzo:=0;
TMR0 := 0;
TRISD:=0;
TRISB := 0xFF;      // Imposta RB come ingresso
TRISC := 0x00;      // Imposta RC come uscita
portC := 0;         // inizializza i valori della porta di uscita
stato_porta:=0;     // inizializza lo stato della porta
TRISB := 0xFF;      // Imposta portB come ingresso
TRISC := 0x00;      // Imposta portC come uscita
portC := 0;         // Azzera uscita portC
INTCON.GIE := 1;    // GIE: Global Interrupt Enable bit
INTCON.INTE := 1;   // INTE: RB0/INT External Interrupt Enable bit
INTCON.INTF := 0;   // INTF: RB0/INT External Interrupt Flag bit
INTCON.PEIE := 0;   // PEIE: Peripheral Interrupt Enable bit
OPTION_REG.INTEDG:=1; //l'interrupt RB0/INT viene scatenato sul fronte di salita
                      // ponendo  OPTION_REG.INTEDG:=1; l'interrupt verrebbe
                      // scatenato sul fronte di discesa
INTCON.RBIE := 1;     // RBIE: RB(4..7) Port Change Interrupt Enable bit
INTCON.TMR0IE := 1;   // TMR0 Interrupt Enable
OPTION_REG.PSA:=1;    // Nessun prescaler applicato
INTCON.RBIE := 1;     // RBIE: RB(4..7) Port Change Interrupt Enable bit
INTCON.TMR0IE := 1;   // TMR0 Interrupt Enable
OPTION_REG.PSA:=1;    // Nessun prescaler applicato
OPTION_REG.T0CS:=0;   // Sorgente per TMR0: Fosc/4
while (true) do
            begin
             if stato_porta=0 then
             portC:=(portC)AND(0x0F)
                                   else
                                   portC:=(portC)OR(stato_porta);
            end;
end.
//_____________________END PROGRAM_____________________
```

Device
Name: P16F877
MCU Clock
Frequency: 8.000000 MHz

Figura 12: Listato MikroPascal PRO for PIC dell'esempio di codice Lezione9_ES6; si faccia riferimento alla Figura 6 per l'implementazione hardware dell'applicazione. Il Timer0 e l'IOC (Interrupt ON-Change) vengono utilizzati per l'implementazione dell'antibouncing firmware.

Esempio 5-Timer come contatore di impulsi

Con gli esempi Lezione9_ES1 e Lezione9_ES2 abbiamo visto come sia possibile utilizzare l'external interrupt RB0/INT per realizzare un contatore. Esiste un altro modo di costruire un contatore mediante interrupt che fa uso del Timer0. Con riferimento allo schema logico-funzionale di Figura 6, si tratta di utilizzare il pin RA4 come sorgente di clock per il timer0. Per semplificare l'esempio si è implementato un contatore a una sola cifra. Volendo implementare un contatore a più cifre è possibile modificare il codice utilizzando oltre al Timer0 per il conteggio un ulteriore timer per scandire il multiplexing dei display a sette segmenti.

In Figura 10 si riporta l'implementazione hardware dell'applicazione di esempio mentre in Figura 11 si riporta l'implementazione firmware (codice MikroPascal).

Si noti in particolare, con riferimento allo schema di Figura 6 e al listato di Figura 11, l'impostazione del bit T0SE per la selezione del segnale presente sul pin RA4 come sorgente di clock del Timer (contatore).

Esempio 6-Timer0 e IOC utilizzati per l'implementazione di un antibouncing firmware

In una delle lezioni precedenti abbiamo parlato di tecniche di antirimbalzo (antibouncing). Esiste una tecnica firmware che in quell'occasione non abbiamo volutamente esposto, che consente di effettuare un efficace antirimbalzo e che si basa sull'impiego di IOC (Interrupt on change) e Timer. In pratica si utilizza come pin di ingresso un pin dotato di interrupt on-change; alla prima transizione dell'ingresso, allo scatenarsi dell'interrupt-on-change si attiva un timer. Le transizioni di segnale costituenti rimbalzi elettrici vengono discriminati mediante l'applicazione di un criterio temporale basato su Timer. Solo dopo che è trascorso un certo intervallo di tempo dal momento in cui si è verificato l'interrupt, si ritiene che i rimbalzi si siano

estinti e le transizioni del segnale vengono nuovamente accettate come valide.
A questo punto il timer viene riazzerato e l'algoritmo è pronto per effettuare una nuova operazione di filtraggio temporale dei rimbalzi. Analizzando il listato di programma (Figura 12) si nota tra le dichiarazioni quella di una costante (si legga il successivo paragrafo per maggiori dettagli in merito alla dichiarazione di costanti) cui è stato dato il nome $T_{_filtro}$.
Variando il valore di questa costante è possibile modificare a piacimento il periodo di filtro dei rimbalzi. Se si utilizza un simulatore come PIC Simulator IDE è possibile verificare come, ponendo per esempio $T_{_filtro}$=10 si abbia un periodo di filtro di circa 10x128us=1,28 ms. In pratica, una volta premuto un pulsante tra RB0 e RB4..RB7, una nuova transizione viene ignorata fino a che non è trascorso un tempo dalla prima transizione proporzionale, secondo un certo fattore, al periodo di filtro T_filtro. Nella pratica, il periodo di filtro sarà pari a 50-100ms e le transizioni che si filtreranno non saranno reali transizioni ma rimbalzi elettrici che scaturiscono dalla pressione del pulsante stesso.
In Figura 7 è riportato il circuito di riferimento per riprodurre fisicamente l'esperimento mentre in Figura 12 si riporta il listato di codice MikroPascal che implementa il metodo. Il tempo massimo di rimbalzo che è possibile sopportare è dato da:

$$T_{max}[us]=T_{_filtro}x256x4x(1/(8MHz))$$

dove T_filtro è la costante che imponiamo all'interno del firmware, 256 è il numero di stati attraverso i quali può evolvere il timer e il 4 è dovuto al fatto che il segnale che sollecita il timer è pari a Fosc/4. Fosc è la frequenza di clock di sistema, in questo caso pari a 8 MHz. Evidentemente, essendo più comodo esprimere il suddetto tempo in millisecondi potremo scrivere:

$$T_{max}[ms]=1000xT_{_filtro}x256x4x(1/8MHz)$$

Nell'esempio mostrato in Figura 12 si è imposto un tempo massimo di rimbalzo pari a 50 ms, motivo per cui dalla formula inversa della precedente si è ottenuto T_Filtro=390 che abbiamo approssimato a 400. Ciò corrisponde al raporto 50ms/128us dove 128 us è la distanza temporale tra un overflow del Timer0 e il successivo.

DICHIARAZIONE E UTILIZZO DI COSTANTI

Approfittiamo dell'occasione fornita dal codice di esempio di Figura 12 per parlare di dichiarazione di costanti. Da un punto di vista pratico, una costante in un codice Pascal è un identificatore opportunamente dichiarato all'interno del codice cui si associa contestualmente tipo e valore. Ai neofiti della programmazione potrebbe sembrare qualcosa di inutile dal momento che si tratta di un identificatore il cui valore non può cambiare durante l'esecuzione. Se il valore è fisso che bisogno c'è di indicarlo con un identificatore? In realtà la definizione di una costane può essere estremamente utile nello stilare un codice di programma ordinato e facilmente manutenibile. E' sufficiente infatti pensare ad una costante il cui impiego è ricorrente nel codice. La sua dichiarazione consente di modificarne il valore accedendo a un solo punto del codice piuttosto che individuandone tutte le ricorrenze al suo interno. Durante la compilazione la costante viene sostituita dal suo valore dichiarato all'interno del codice, di conseguenza il valore viene a far parte a tutti gli effetti del codice oggetto senza comportare consumo di memoria RAM come accade invece per le variabili. La costate può essere utilizzata in più punti del codice ma può essere dichiarata una sola volta proprio a causa del concetto stesso di costante. Essa viene dichiarata nella sezione dichiarativa mediante la seguente sintassi che prevede l'impiego della parola chiave *const*:

const *constant_name* [: *type*] = *value*;

E' uso comune utilizzare per le costanti identificatori espressi in lettera minuscola. Mentre il valore deve sempre essere definito, il tipo del dato può anche essere omesso. In questo caso il compilatore assume come tipo quello di dimensioni minime possibili in grado di contenere il dato stesso. Questo non è vero nel caso in cui si debba dichiarare come costante un array. MikroPascal inoltre, come per le variabili, consente la dichiarazione di più costanti attraverso una sola dichiarazione rendendo il codice più compatto. Di seguito si riporta un esempio di dichiarazione di costanti tratto dall'Help in linea di MikroPascal PRO for PIC 5.61.

const
  MAX : longint = 10000;
  MIN = 1000; *// il compilatore assume come tipo la word*
  SWITCH = 'n'; *// il compilatore assume come tip il char*
  MSG = 'Hello'; *// il compilatore assume come tipo la stringa*
  MONTHS : **array**[1..12] **of** byte = (31,28,31,30,31,30,31,31,30,31,30,31);

OSSERVAZIONI E APPROFONDIMENTI

Gli esempi mostrati potevano essere scritti anche in forma più compatta senza referenziare i singoli bit dei registri di configurazione degli interrupt in maniera esplicita. Riteniamo tuttavia opportuno referenziare questi in maniera esplicita perché questo modo di agire riduce la possibilità di commettere errori e rende più semplice e agevole successivamente interpretare il codice stesso. Questo appare cioè più leggibile sebbene stilisticamente meno compatto.
Il fatto di scrivere esplicitamente i bit di configurazione dei Timer e degli interrupt facendo riferimento agli acronimi degli stessi e dei registri cui appartengono aiuta il programmatore a ridurre la probabilità di commettere degli errori dal momento che l'acronimo aiuta ad associare in maniera naturale agli stessi bit il loro significato. Ovviamente, è comunque anche possibile eseguire con un'unica assegnazione la configurazione dell'intero registro, per esempio, all'intero registro OPTION_REG. Questo secondo modo di operare presenta fondamentalmente un unico vantaggio: la maggiore compattezza del codice. E' in fine importante osservare che a essere rigorosi sarebbe opportuno integrare l'antirimbalzo in tutti gli esempi mostrati in questa lezione.

Per approfondire

[1] PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers (www.mikrochip.com)
[2] Help in linea MikroPascal Pro for PIC v. 5.61
[3] MikroPascal PRO for PIC v.5.61 demo e documentazione relativa (www.mikroe.com)
[4] Corso MikroPascal PRO for PIC -Parte 7 (Fare Elettronica n.333)-Interrupt e Timer-Parte I
[5] Corso MikroPascal PRO for PIC -Parte 8 (Fare Elettronica n.334)-La gestione dei display a 7 segmenti

CONCLUSIONI

In questa lezione abbiamo mostrato esempi e applicazioni pratiche di alcuni degli interrupt più comunemente utilizzati nell'impiego dei controllori e dei controllori PIC in particolare. Si tratta degli interrupt:

- external interrupt RB0;
- interrupt on change (RB4..RB7);
- Timer0.

Non è difficile modificare gli esempi mostrati in maniera da adattarli a scopi specifici. Per quanto concerne l'impiego dei Timer abbiamo preso a riferimento il Timer0, ma l'implementazione di applicazioni firmware basate su Timer1 e Timer2 del PIC16F877 o di altri timer di altri modelli di controllori PIC non è molto differente.
Abbiamo mostrato anche come sia possibile utilizzare l'interrupt Timer0 come contatore di fronti d'onda che si presentino sul pin RA4 utilizzato come ingresso. In queste condizioni il segnale posto su RA4 viene utilizzato al posto del clock interno.
Abbiamo anche mostrato come sia possibile gestire più interrupt all'interno dello stesso codice di programma.
Esistono numerosi altri tipi di interrupt integrati nei controllori PIC ma il loro impiego non è molto differente da quanto abbiamo avuto modo di vedere per gli interrupt presi in considerazione in questa lezione. Interrupt legati a conversioni ADC, comunicazione USART, gestione di EEPROM verranno presi in considerazione al momento opportuno in lezioni successive in concomitanza con la trattazione di argomenti specifici.
Gli esempi mostrati, specie quelli che consentono il conteggio degli impulsi che giungono sul pin RB0 oppure sul pin RA4 del controllore possono essere facilmente utilizzati per implementare dei contaimpulsi oppure dei contapezzi o contagiri. Allo scopo è sufficiente utilizzare delle fotocellule oppure dei fotodiodi per trasformare il passaggio dei pezzi presenti su un nastro trasportatore o la rotazione di un asse in un treno di impulsi che contati dal controllore possono contestualmente essere rappresentati su display. Il conteggio del numero di impulsi di un segnale consente inoltre di misurarne indirettamente la frequenza.
Si consiglia, al fine di acquisire la massima padronanza possibile con gli interrupt di:

- analizzare attentamente i listati di codice presentati alla luce di quanto esposto nelle precedenti due lezioni e dei relativi schemi elettrici;
- di ripetere gli esperimenti in hardware e firmware o solo in firmware attraverso l'impiego di simulatori quali quelli segnalati nella lezione 7 (in particolare PicSimLab e PC Simulator IDE);
- di provare a modificare i listati stessi adattandoli ad applicazioni specifiche.

Nella prossima lezione cercheremo di utilizzare quanto fin qui appreso su I/O, gestione di display LCD e interrupt per implementare in firmware un Timer programmabile o un allarme temporizzato. Costruiremo in tal modo un'utile applicazione che potrà essere successivamente arricchita di altre funzionalità.

di GIOVANNI DI MARIA

DOMOTICA CON RASPBERRY PI

In queste pagine vedremo come realizzare una applicazione, sempre richiesta dai progettisti. Si tratta di piccolo sistema di domotica che utilizza il Raspberry Pi, con il quale si possono comandare fino a cinque diverse utenze, in modo del tutto indipendente

Figura 1: Il sistema al lavoro

Ecco un piccolo maggiordomo ubbidiente e di fiducia, che possiamo tranquillamente lasciare a casa. Con esso possiamo comandare fino a cinque diverse utenze anche alimentate a 230V con corrente massima di 4A.

UN SISTEMA DI DOMOTICA

L'uomo tende, per sua natura, ad automatizzare tutte le sue attività. Anche la sua abitazione non è esente da questo destino. Ventilatori, televisori, stereo, sistemi di irrigazione e tanto altro possono essere comandati automaticamente dal computer, anche in assenza dell'uomo.
Oltre ad un piacere intrinseco e ad una sensazione di potere, la domotica è molto utile in tante situazioni. Con essa infatti si automatizza parte dell'impianto elettrico di casa, in modo che le utenze possano essere attivate o disattivate ad una certa ora.

CARATTERISTICHE

Il sistema proposto in queste pagine costituisce un prototipo di base, per iniziare a sperimentare questa categoria di applicazioni, con il Raspberry Pi. Con esso si possono comandare fino a cinque utenze, tramite la tastiera e una normale televisione (anche in B/N).
Dal momento che il pilotaggio avviene attraverso cinque relè meccanici, le utenze possono funzionare con qualsiasi tensione, anche quella di rete casalinga a 230V. L'importante è non superare i 4 A di corrente.
Ecco, in dettaglio, alcune caratteristiche salienti del sistema di domotica e le sue prerogative e mansioni base:

- Pilotaggio fino a 5 utenze;
- Attivazione e disattivazione manuale delle utenze, tramite tastiera;
- Programmazione oraria (ora di inizio e di fine) delle utenze;
- Possibilità di attivare o meno la programmazione oraria;
- Visualizzazione in tempo reale dello "status" delle utenze;
- Visualizzazione in tempo reale dell'orario corrente.

SCHEMA ELETTRICO

Lo schema elettrico si compone di due unità:

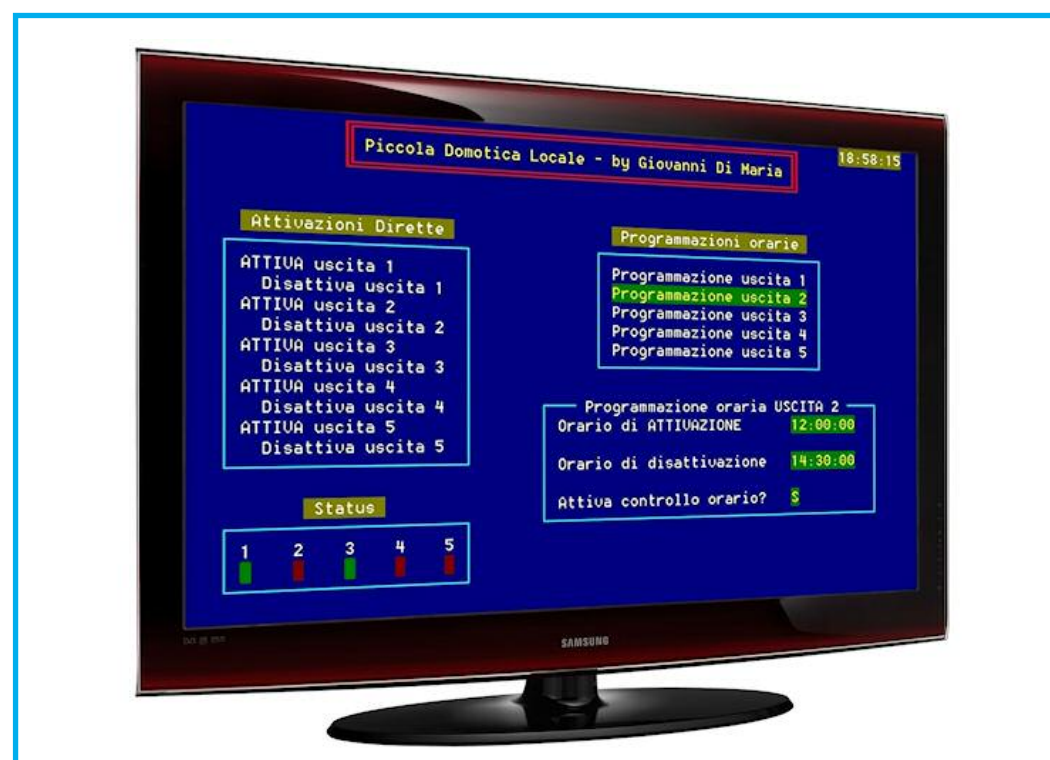
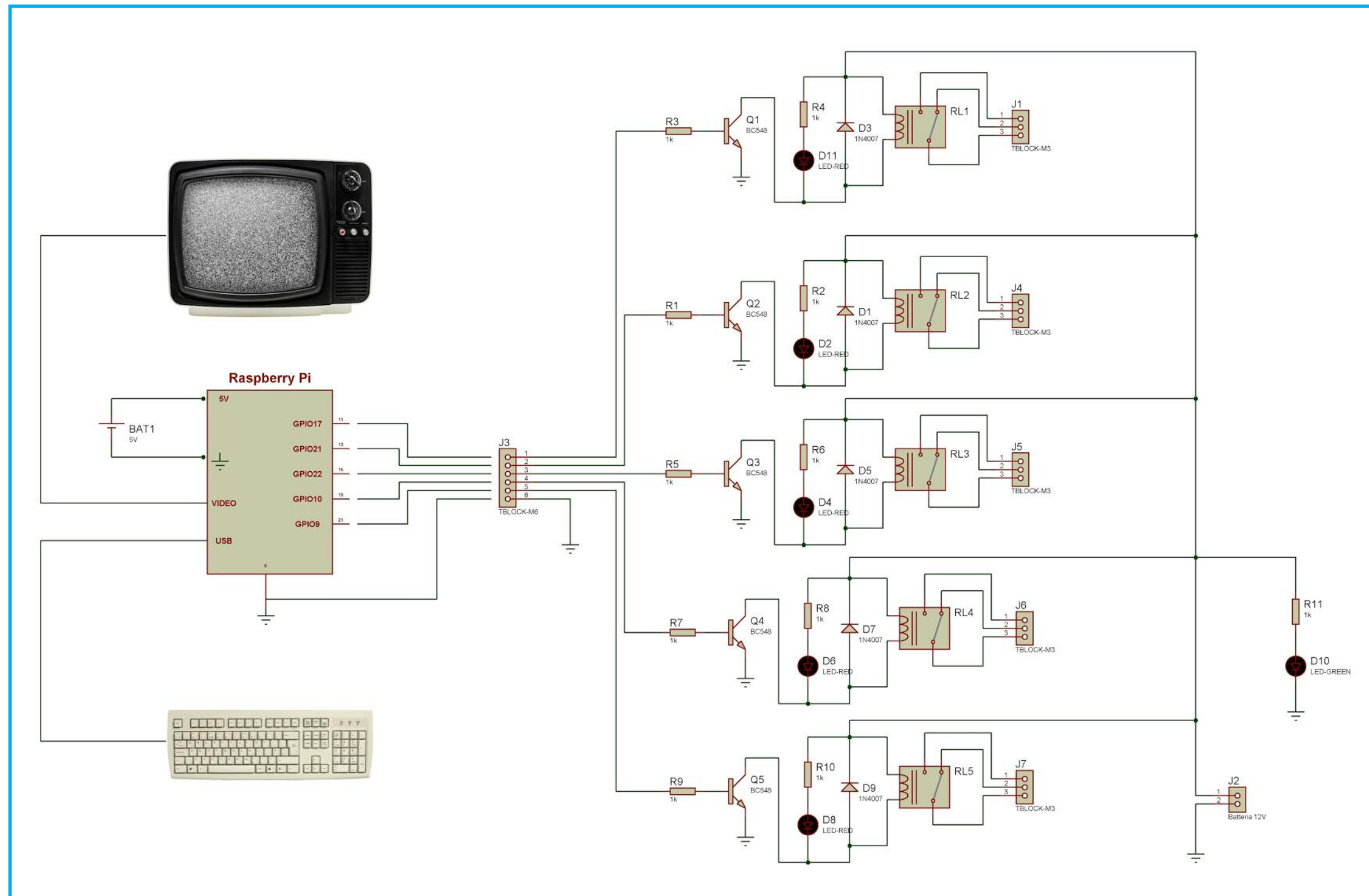



Figura 2: Schema elettrico

- L'unità intelligente, rappresentata dal Raspberry Pi;
- L'unità di pilotaggio e di bassa potenza, rappresentata dalla scheda dei relè.

Il Raspberry Pi (con il suo software) ha il compito di pilotare le cinque uscite, che avviene in logica digitale, a 3.3V (livello alto) e 0V (livello basso). Tale tensione, insufficiente per comandare i relè, viene potenziata da altrettanti transistor.

Guardando lo schema elettrico generale, ma anche focalizzandoci sullo schema del singolo stadio, possiamo notare che l'uscita digitale del Raspberry Pi è collegata ad un transistor della scheda, tramite la resistenza di limitazione della corrente di base. Sul collettore del transistor troviamo due carichi: il diodo led, con la relativa resistenza di limitazione, che serve per testimoniare o meno l'attivazione di una utenza e, soprattutto, il relè che ha il compito di pilotare l'utenza vera e propria. Dal momento che la presenza della bobina potrebbe innescare pericolose veloci extratensioni, nocive per il transitor, è stato inserito un diodo in anti-parallelo all'induttore, che smorza i picchi di tensione elevati. I tre terminali dei contatti del relè sono collegati ad altrettanti terminali dei morsetti, affinché siano montati e collegati sul carico desiderato. Fate attenzione che ogni relè ha un terminale comune, uno normalmente aperto (NA) ed uno normalmente chiuso (NC).

La scheda non è optoisolata ed il contatto con il Raspberry Pi è diretto, attraverso le resistenze di limitazione delle basi dei transistor.

Occorre assicurarsi che le masse del Raspberry Pi e della scheda elettronica devono essere collegate tra loro.

IL PCB

Per la costruzione della scheda, occorre realizzare il circuito stampato, la cui traccia, in scala naturale, è riportata in figura. Esso risulta abbastanza semplice, anche se si consiglia vivamente di utilizzare la tecnica della fotoincisione, per ottenere un risultato più affidabile e professionale. Una volta che la basetta è pronta, occorre praticare dei fori con punta di trapano da 0,8 mm oppure da 1 mm in corrispondenza delle piazzole. Si consiglia di irrobustire le pista di rame che fanno capo ai relè, con una stagnatura massiccia. Questo allo scopo di prevenire un surriscaldamento eccessivo del circuito, se si utilizzano grossi carichi (con potenza maggiore di 600 W).

MONTAGGIO E ASSEMBLAGGIO

Si può quindi procedere alla saldatura dei componenti, iniziando da quelli a basso profilo, quali resistenze e diodi per poi proseguire con quelli più ingombranti, come i morsetti, i transistor e i led. Occorre (non ci stancheremo mai di ripeterlo) prestare molta attenzione ai componenti polarizzati. Per la saldatura utilizzate un piccolo sal-

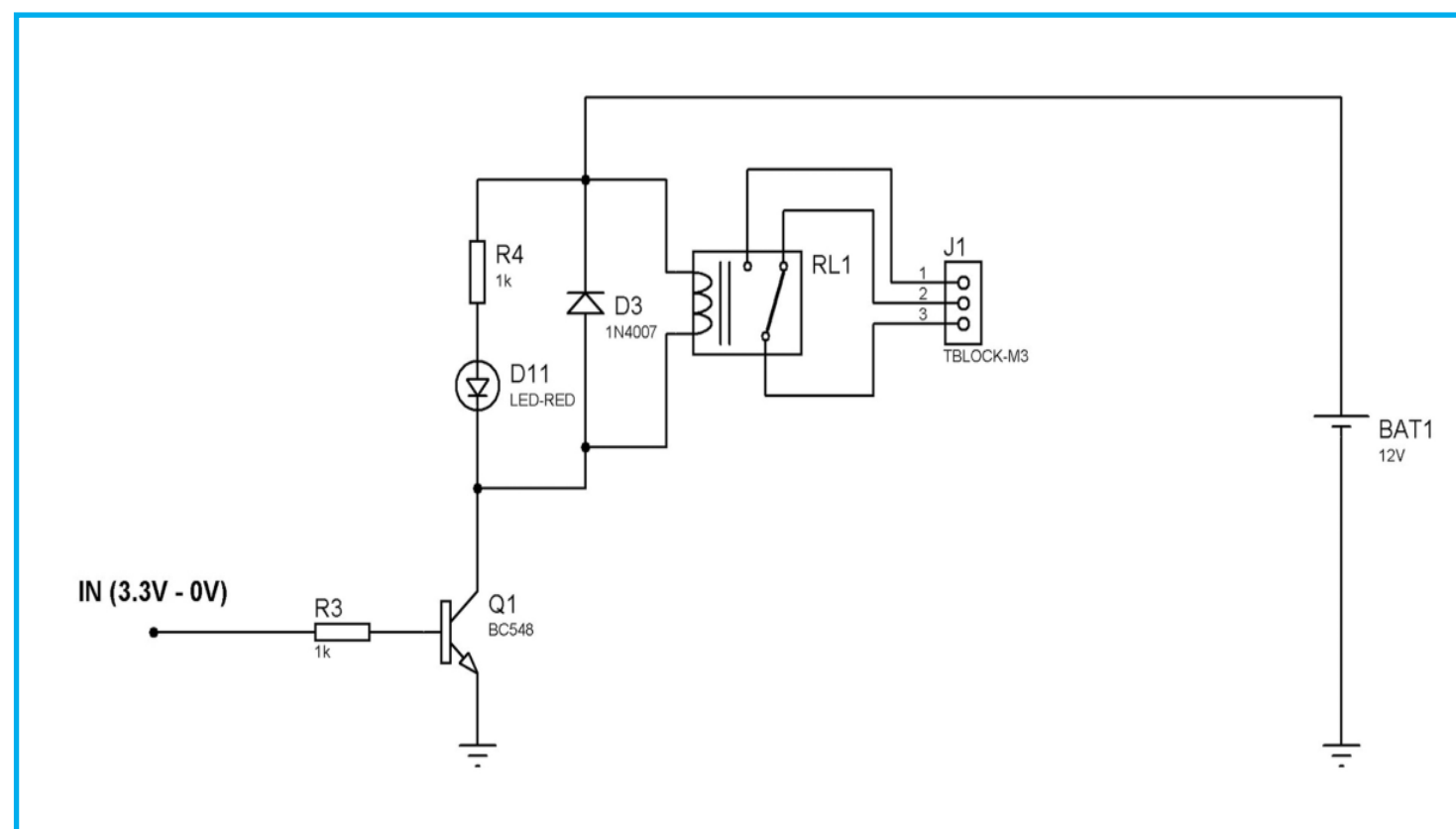



Figura 3: Schema di principio di un singolo stadio

datore della potenza di circa 30 W, facendo attenzione a non surriscaldare i componenti elettronici, che non sopportano un eccesso di calore.

CONNESSIONI ELETTRICHE

Bene, è arrivato il momento di provare il sistema. Prima di collegare la scheda dei relè al Raspberry Pi, consigliamo vivamente di provarla in modo indipendente; non si sa mai, l'errore è sempre in agguato. Per eseguire un collaudo preliminare, occorre alimentare la scheda con una tensione di 12V e, con una batteria da 3V occorre fornire tensione ad ogni singolo ingresso digitale. Se tutto è a posto, dovrebbe sentirsi i "tic tac" dei relè con la relativa illuminazione dei diodi led. Per disattivare il contatto occorre invece staccare la batteria ed, eventualmente, collegare l'ingresso del morsetto a massa.
A questo punto si può effettuare il collegamento elettrico, come chiaramente mostrano le illustrazioni in queste pagine.

IL SOFTWARE

Il software applicativo è estremamente semplice e, se vogliamo, ridotto all'osso. Con esso si possono gestire e programmare 5 utenze diverse ed indipendenti. La videata è unica ed è composta dalle seguenti sezioni:

1) Titolo
2) Orologio
3) Attivazioni dirette
4) Status
5) Programmazioni orarie

1-Titolo

E' ovviamente l'intestazione del programma, che contiene anche il nome dell'autore.

2-Orologio

Si tratta di un piccolo orologio in tempo

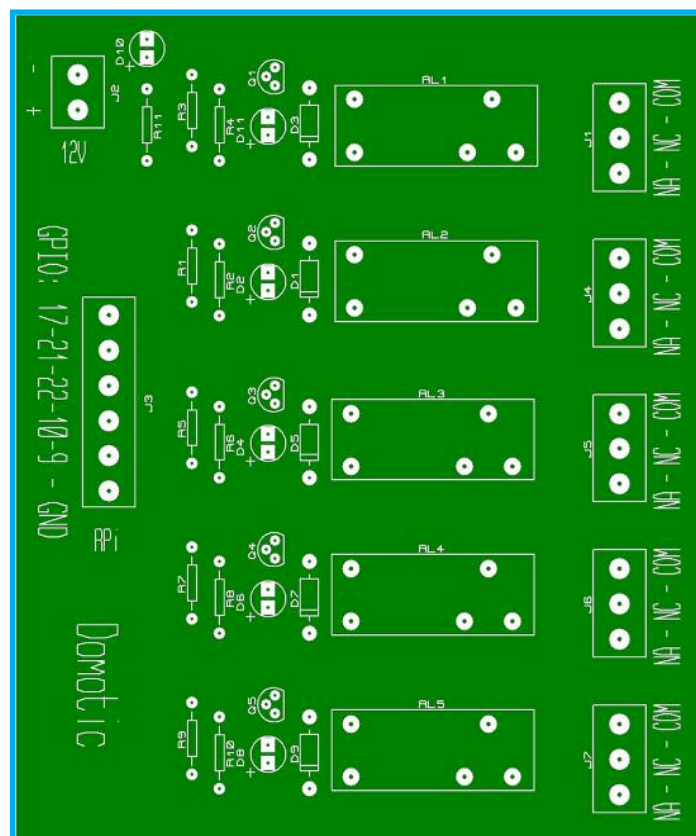



Figura 4: Disposizione dei componenti

Figura 5: Circuito stampato (lato rame)

reale, molto comodo, localizzato in alto a destra dello schermo TV.

3-Attivazioni dirette

Tramite questo menù, situato a sinistra, è possibile attivare e disattivare manualmente una o più uscite della scheda. E' molto utile in caso di test delle periferiche. Le utenze si scelgono attraverso il menù a discesa, con le frecce cursore della tastiera USB.

4-Status

E' un piccolo riquadro che mostra il numero dell'utenza assieme al suo stato logico: attivata (colore verde) oppure disattivata (colore rosso).

5.Programmazioni orarie

E' un altro menu a discesa, situato sulla destra, che permette di scegliere una utenza, al fine di programmarla per l'automazione. La scelta dell'uscita visualizza un riquadro, che attende l'immissione dell'orario di attivazione, dell'orario di disattivazione e se tali orari devono essere considerati o meno dal programma.

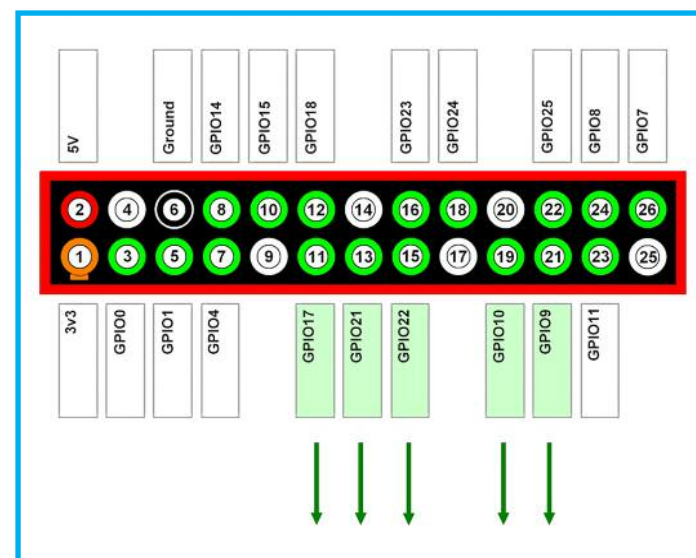



Figura 6: Le porte di output del Raspberry Pi utilizzate

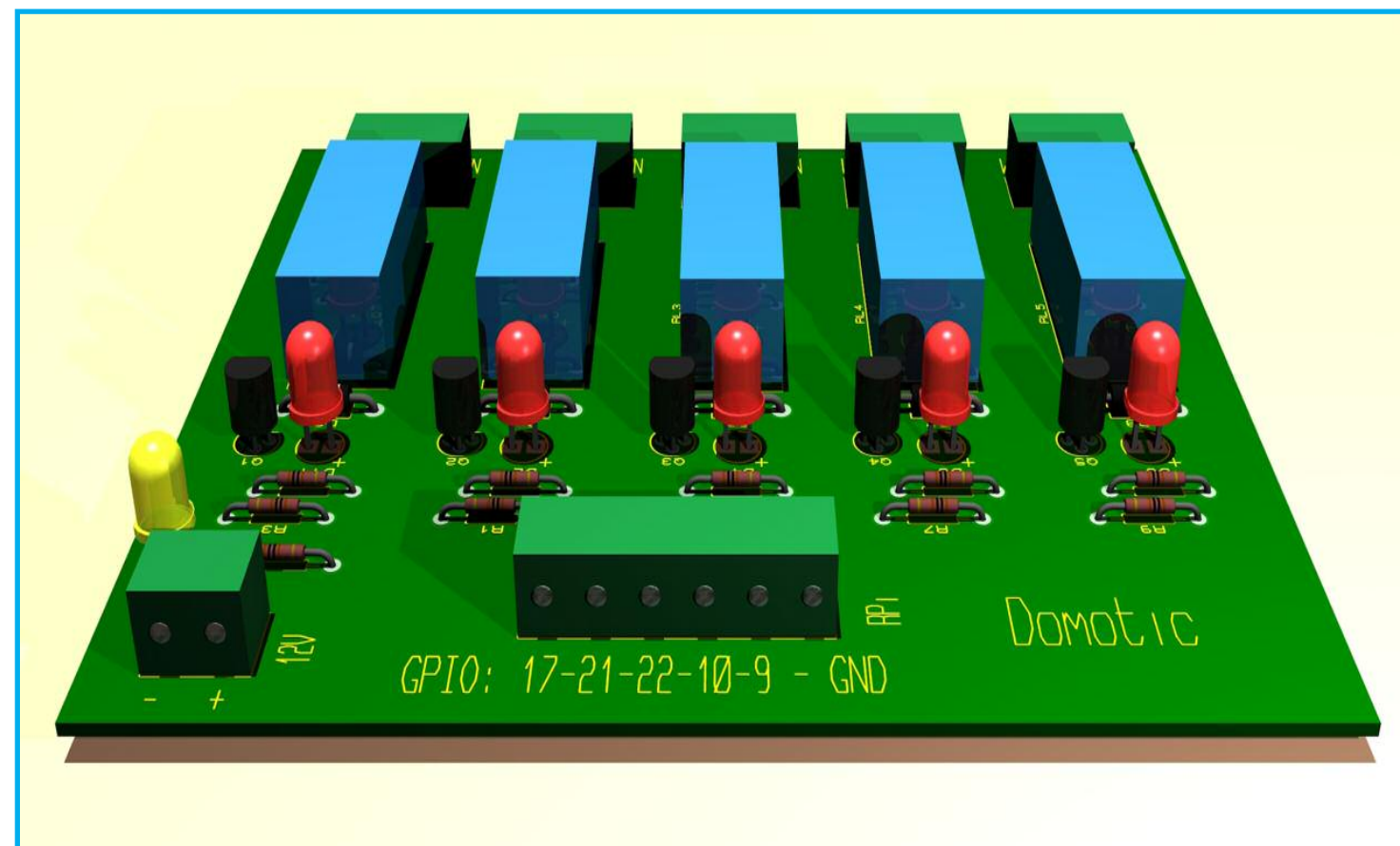



Figura 7: Vista frontale 3D del prototipo della scheda

Figura 8: Il cablaggio dell'intero sistema

Figura 9: La videata del programma

Figura 10: Pinout del transistor BC548

Il software è liberamente scaricabile dal sito di Fare Elettronica oppure all'indirizzo http://www.elektrosoft.it/download.asp. Esso è un file eseguibile, pertanto basta copiarlo in qualsiasi directory del sistema Linux del Raspberry (si consiglia la directory /usr/bin) ed eseguirlo con il comando:

```
./domloc
```

Si ricorda altresì che per il funzionamento del sistema occorre essere avere l'accesso come *root*.

CONCLUSIONI

Il sistema di domotica darà sicuramente tanta utilità e soddisfazione a quanti lo realizzeranno. I campi di applicazione sono veramente tanti: programmare l'accensione e lo spegnimento di luci, scaldabagni, lavatrici, attivazione irrigazione automatica, ecc.

Una sola raccomandazione: il Raspberry Pi dispone di un orologio interno gestito dal sistema operativo. Lo si deve regolare correttamente prima di avviare l'"applicazione. Se si toglie l'alimentazione, l'orologio si resetta, pertanto utilizzate batterie tampone al fine di prevenire eventuali black-out.

Alle prossime, con altri progetti per Raspberry Pi.

Figura 11: Pinout diodo Led

Figura 12: Pinout diodo 1N4007

Elenco componenti

R1 ~ R11	1 KΩ 1/4 W
Q1 ~ Q5	Transistor BC548 o equivalenti
D1,D3,D5,D7,D9	Diodi 1N4007
D2,D4,D6,D8,D11	Diodi Led Rossi
D10	Diodo Led Verde
BAT1	Alimentazione 5V – 800mA
J1,J4 ~ J7	Morsetto a 3 posti per c.s.
J2	Morsetto a 2 posti per c.s. (alimentaz. 12V)
J3	Morsetto a 6 posti per c.s. (oppure 2 da 3 posti)
RL1 ~ RL5	Relè a 12V
	1 Televisore
	1 Tastiera per PC USB
1 Raspberry (mod. A o B)	
	Cavetterie varie
	Alimentazione 12V - 500mA

Figura 13: Il modello di relè utilizzato

di GIORGIO OBER

INTERFACCIAMENTO DEI PROCESSORI

GESTIONE KEYPAD CON KEY ENCODERS

(parte 18ª)

Il numero di linee di I/O necessarie per leggere una Keypad può essere reso minimo con l'aiuto di dispositivi combinatori intermedi, ma esistono pregiati integrati specializzati in questo lavoro: di essi ci occuperemo in questa puntata

La scorsa puntata abbiamo visto che una tastiera con 16 pulsanti indipendenti può essere gestita con sole 5 linee di microcontrollore, se interfacciata con dispositivi in grado di organizzarne la lettura sequenziale dei contatti (come un Data Selector/Multiplexer 74150) o di rilevarne la presenza e codificarne la posizione (come un Priority Encoder 74LS148); con un tastierino a Matrice (una Keypad con 16 tasti, disposti sugli incroci delle sue 4 Righe e delle sue 4 Colonne) affidato al controllo diretto del single-chip (cioè senza alcuna interfaccia) ne sono state necessarie 8, metà programmate in ingresso e metà in uscita.

Figura 1 - 16-Key Encoder 74C922: pinout

Se le risorse di I/O a disposizione sono limitate è opportuno far ricorso a circuiti integrati progettati specificatamente per questo tipo di servizio; tra essi i CMOS 16-Key Encoder 74C922 e 20-Key Encoder 74C923, in grado di funzionare in un'ampia gamma di alimentazione (da 3V a 15V) con basso consumo di energia (da 1,1 mA a 2,6 mA); il loro pinout è mostrato rispettivamente in **Figura 1** e in **Figura 2**.

La loro architettura è in grado di fornire tutta la logica combinatoria necessaria sia per la scansione di una Matrice di contatti a una via (SPST, *Single Pole, Single Throw*) che per la piena decodifica dei suoi tasti: in cambio di una piccola spesa possiamo disporre di un vero specialista; la **Figura 3** mostra lo schema funzionale dell'Encoder a 16 tasti 74C922 (sostanzialmente identico

Figura 2 - 20-Key Encoder 74C923: pinout

al 74C923, se non per il fatto di adattare le comuni risorse a 16 invece che a 20 tasti). Ricordiamo il fenomeno di *Key Ghosting*, tipico delle tastiere a Matrice e di quelle economiche (contenenti solo tasti, senza altro hardware); esso si può manifestare quando 3 tasti sono premuti contemporaneamente su tre angoli del quadrangolo identificato da due Righe e da Due Colonne: in questa situazione, a causa del percorso che la corrente può seguire attraverso i contatti effettivamente chiusi, è ritenuto premuto anche il contatto del pulsante (per questo detto "fantasma") che occupa l'angolo rimanente, anche se il tasto stesso è ancora aperto.

Di questo abbiamo parlato ampiamente sul numero precedente, arrivando alla conclusione che, per risolvere ogni problema, basta dotare ogni tasto di un diodo in serie, così da impedire il passaggio di corrente tra tasti adiacenti; la sofisticata "*Encoding Logic and 2-key Rollover*" di entrambi i dispositivi è in grado di eliminare il fenomeno dei contatti fantasma senza dover inserire alcun diodo.

È presente anche un circuito (*Key Bounce Elimination*) per il controllo del rimbalzo del tasto premuto, un fastidioso evento sempre da mettere in conto: il contatto può rimbalzare più volte, ma se rimane basso al termine del periodo di antirimbalzo (*Debounce Time*, regolato dalla capacità del condensatore aggiunto dall'esterno) la sua chiusura si ritiene valida.

La velocità di scansione del tastierino può essere fornita applicando al piedino d'ingresso OSC un preciso clock esterno, ma anche collegandovi un condensatore che, in virtù della presenza interna di un oscillatore astabile (realizzato con una porta invertente Trigger di Schmitt con in parallelo un resistore da 10k) assicura una frequenza di oscillazione sufficiente per garantire il servizio richiesto, anche se poco precisa e poco stabile; il suo valore dipende da quello della tensione di alimentazione, dalla tensione di soglia superiore e inferiore dell'inverter e dalla temperatura ambiente, e può essere calcolato con la formula riportata nello schema funzionale; con Vcc=5V può essere sintetizzata in **f = 1,22/R*C**.

Nelle Application Notes dei datasheet si raccomanda di usare per il condensatore KBM una capacità 10 volte più grande di quella del condensatore su OSC; più precisamente si consigliano rispettivamente i valori di 1 µF e di 100 nF: il primo assicura un Debounce 10 ms e il secondo (con il resistore interno da 10 kohm) fissa a 1220 Hz la frequenza dell'oscillatore.

Poiché i tastierini disponibili hanno specifiche molto personali non è da scartare l'idea di fare delle prove con diversi valori di capacità, tenendo conto che valori più grandi aumentano il tempo di Debounce e diminuiscono la frequenza di scansione.

Dopo aver predisposto i due condensatori i nostri dispositivi sono operativi: la frequenza dell'oscillatore interno viene applicata ad un contatore binario modulo 4, cioè chiamato a generare 4 combinazioni a 2 bit (00, 01, 10 e 11), subito spedite in uscita per costituire la parte meno signifi-

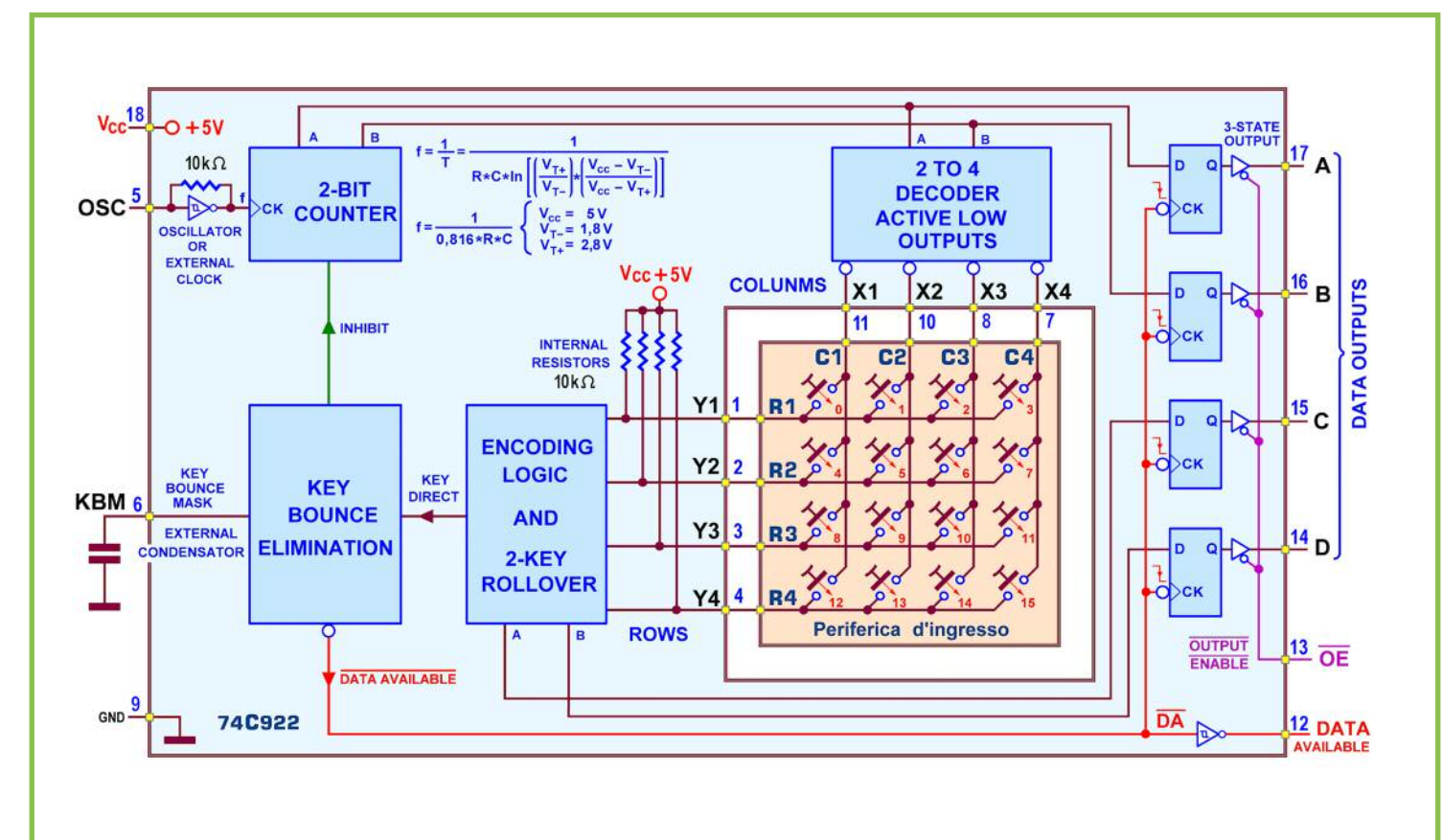



Figura 3 - 16-Key Encoder 74C922: schema funzionale

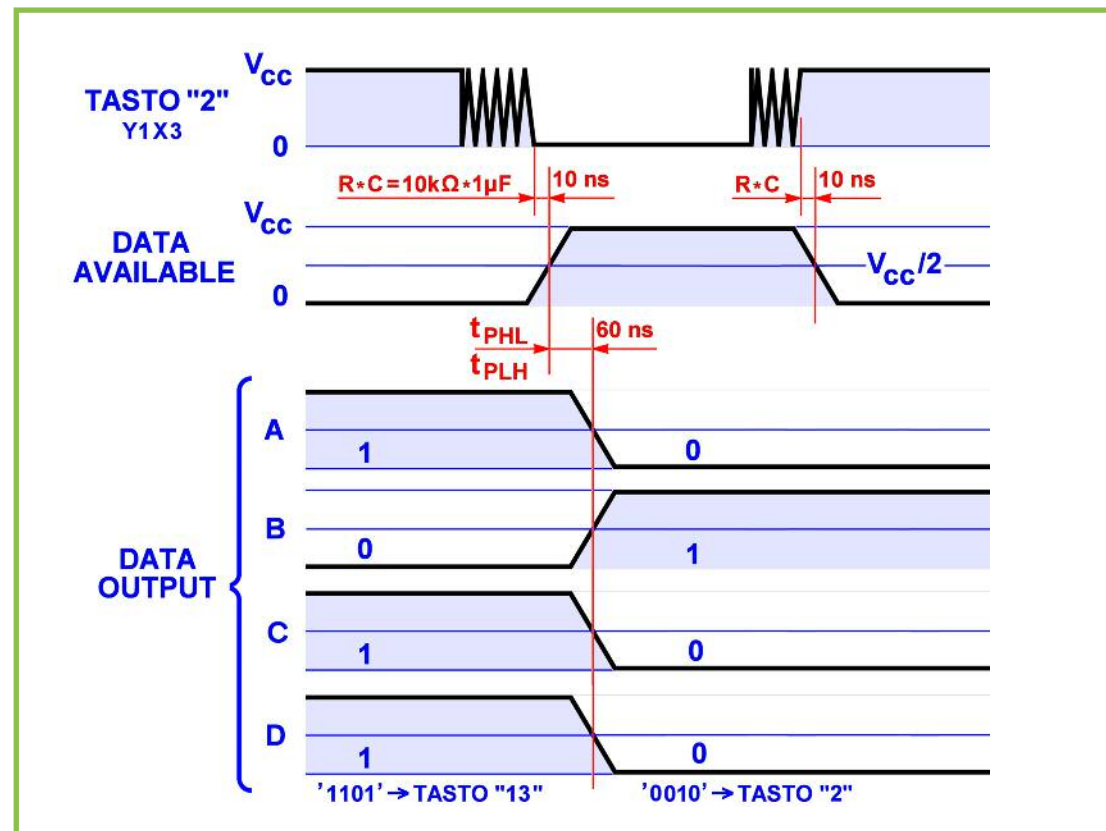



Figura 4 - 16-Key Encoder 74C922: diagramma temporale

cativa del codice identificativo finale da associare al tasto premuto correttamente; la coppia di bit viene applicata anche agli ingressi di un decoder binario "da 2 a 4" che pone a 0 l'uscita ad essa corrispondente (lasciando a 1 le altre tre).
In questo modo porta a livello basso (una dopo l'altra) le Colonne da X1 a X4, rendendo possibile il rilievo della pressione dei 4 tasti di ciascuna Riga; gli ingressi del circuito di codifica (*Encoding logic*) sono collegati alle 4 Righe (da Y1 a Y4) della Keypad e, se nessun tasto è premuto, sono tenuti a 1 logico dal rispettivo resistore di pull-up interno.
Quando invece si preme un tasto, lo 0 logico presente sulla Colonna ad esso relativa porta basso anche il livello della Riga corrispondente; in questo preciso istante l'encoder blocca l'ingresso dalle altre Righe Yn e (tramite la linea *Key direct*) spedisce un segnale di sincronismo che fa partire la temporizzazione del circuito di eliminazione rimbalzi (*Key Bounce Elimination*); esso blocca (*Inhibit*) il conteggio del contatore che, di conseguenza, obbliga il decoder a mantenere costantemente selezionata anche la Colonna del tasto premuto.
In questo modo viene assicurata anche la funzione *2-Key Rollover*: essendo bloccate tutte le scansioni ed essendo disabilitata la lettura da tutti gli altri ingressi Y di Riga, la pressione di un secondo tasto non verrà riconosciuta fino a quando il circuito che monitorizza i rimbalzi non verrà reinizializzato, cosa che avverrà solo quando il tasto che ha dato inizio alla procedura verrà rilasciato.
La situazione di stallo permane per tutto il tempo di Debounce, stabilito dal valore del condensatore su KBM; se la pressione del tasto provoca un rimbalzo il segnale d'ingresso sulla Riga passerà di nuovo da 1 a 0 e la temporizzazione verrà fatta ripartire; la chiusura verrà ritenuta valida solo se il contatto rimane basso per un intero periodo di antirimbalzo senza subire ulteriori picchi positivi.
Non appena la logica di antirimbalzo ritiene chiuso il contatto il circuito di controllo viene resettato (va in *timeout*) e forza bassa la linea *Data Available*, il fronte di discesa della quale opera la memorizzazione (sulle uscite dei 4 Flip-Flops D-Types) del nibble che esprime, in binario, il numero d'ordine della sua posizione, da 0000 a 1111; in particolare i due bit meno significativi rappresentano il numero della Colonna (da X1=00 a X4=11, uscita del contatore e ingresso del decoder) e i due bit più significativi quello della Riga (da Y1=00 a Y4=11, uscita del encoder) su cui è posto il tasto.
Naturalmente il tasto rimbalzerà anche quando viene rilasciato; per evitare che l'encoder interpreti questo rimbalzo come dovuto ad una ulteriore chiusura dello stesso contatto, farà ripartire il timer del circuito *Key Bounce Elimination* che, come prima, consumerà più volte il tempo di debounce, fino a quando il tasto verrà ritenuto stabilmente aperto; solo ora la linea *Data Available* viene riportata bassa, il contatore sarà riavviato e il decoder riattiverà la scansione delle Colonne sulle sue uscite. Prima di questo istante nessun'altra chiusura potrà essere riconosciuta (a conferma della disponibilità della funzione *2-Key Rollover*); le linee *Data Output* continueranno a mantenere memoria del codice dell'ultimo tasto trovato premuto, anche dopo che esso è stato rilasciato; e, non appena ne viene premuto uno nuovo, partirà da capo tutto il processo appena descritto.
Da notare che il segnale DA che opera la memorizzazione del dato YnXn viene reso disponibile, invertito, sull'uscita (attiva alta) *Data Available*, per poter avvisare i circuiti esterni che è stata rilevata la pressione valida di un contatto; le uscite interne della memoria sono governate da altrettanti buffer 3-state non invertenti e rese disponibili sulle *Data Outputs* quando l'ingresso Output Enable è basso; la possibilità di forzare le uscite esterne in alta impedenza (con OE=1) rende possibile il loro utilizzo in strutture di tipo Bus.
Il diagramma temporale di **Figura 4** (rielaborato da quello del datasheet) riassume graficamente le considerazioni appena descritte; il tempo di Debounce è T=RC e vale 10 ms, se il condensatore posto sul pin KBM ha capacità C=1 μF (essendo R=10Kohm); se le *Data Outputs* sono abilitate (con OE=0) l'informazione relativa al tasto rilevato premuto è disponibile dopo un breve ritardo di propagazione, tpd, uguale al più grande dei due tempi tPLH (low-to-high propagation time) e tPHL (high-to-low propagation time); nel nostro caso è variabile con il valore della tensione di alimentazione e, con Vcc=5V, vale tpd=tPLH=PHL=60 ns.
La **Figura 5** mostra lo schema per interfacciare una Keypad a 16 tasti con il nostro Key Encoder 74C922, nelle condizioni suggerite nella descrizione, cioè tempo di Debounce di 10 ms e frequenza di scansione dei tasti pari a 1220 Hz; il microcontrollore si limiterà a rimanere in attesa di un livello alto sul bit4 di PortA e, per poter capire quale tasto è stato premuto, potrà leggere con comodo il nibble sui bit meno significativi della stessa porta, mantenuto inalterato fino all'arrivo di un nuovo suc-

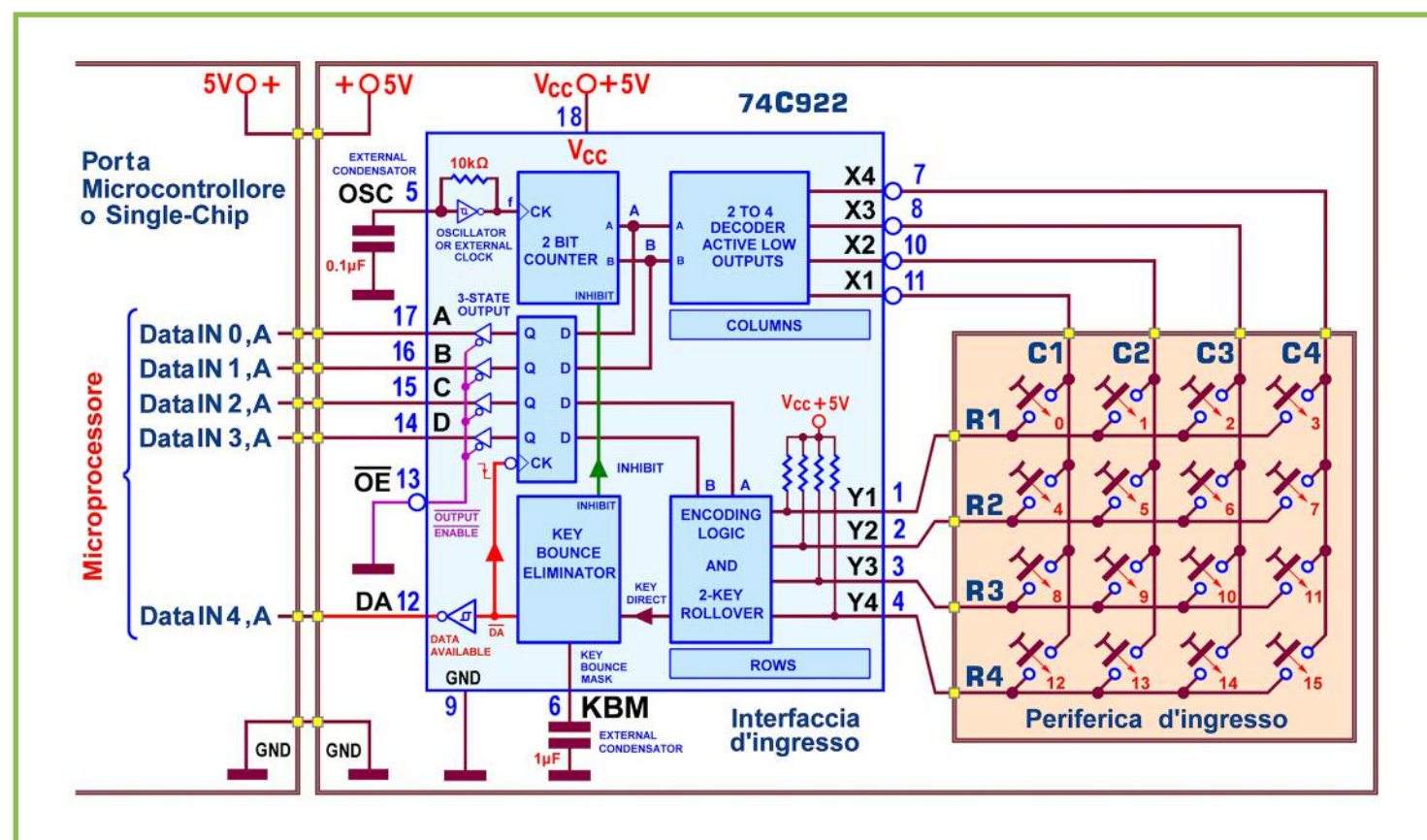



Figura 5 - Lettura di una 16 Button Keypad con Key Encoder 74C922 via PORTA di un microcontrollore

cessivo impulso sul bit4.

Naturalmente è possibile risparmiare una linea d'ingresso collegando l'uscita *Data Available* alla linea di Interrupt della CPU del computer o del micro: in questo modo si evita anche la necessità di obbligare la CPU a consumare tempo in attesa (in polling) della pressione del contatto, garantendo il servizio solo quando questo evento si manifesta.

Se dovesse esser necessario rilevare l'istante in cui il tasto memorizzato viene rilasciato è sufficiente attendere che la linea collegata sul bit 4 torni a zero.

In **Figura 6** viene proposta la soluzione per gestire due Keypads a 16 tasti con un unico Key Encoder 74C922; in condizioni d'attesa le 4 Colonne (da X1 a X4, in comune ad entrambi i tastierini) sono tutte a 1 logico meno una, portata in sequenza a livello basso.

Le 4 linee d'ingresso dell'integrato (da Y1 a Y4) sono tenute a 1 logico da un resistore di pull-up interno e sono collegate alle 4 Righe delle due Keypads, in parallelo tra loro, direttamente a quelle della prima e tramite un diodo a quelle della seconda, ciascuna tenuta a 1 da un resistore esterno da 100 kOhm (valore suggerito dal stasheet).

Quando un tasto viene premuto lo 0 presente sulla sua Colonna porta basso anche il livello della sua Riga e, se essa appartiene al tastierino con tasti da 1 a 15, anche la linea d'ingresso Yn viene direttamente forzata a 0; se invece appartiene al tastierino con tasti da 16 a 31, ci penserà il diodo, passando in forte conduzione, ad imporre alla linea d'ingresso Yn la sua caduta di tensione, portandola ad un livello compatibile con lo 0 logico.

In ogni caso sappiamo che, per tutto il tempo durante il quale l'uscita *Data Available* è trovata ad 1, la sofisticata architettura del 74C922 assicura ad entrambi i tastierini un trattamento *2-Key Rollover*, sicuro anche in presenza degli inevitabili rimbalzi (*Keys Bounce*) dei tasti, sia in chiusura che in apertura, il codice binario del tasto premuto (e rilasciato) è ora formato dai 4 bit garantiti dalle *Data Outputs* e dal valore (associato al bit più significativo) dell'uscita della NAND a 4 ingressi 74C20 (che passa ad 1 solo se almeno uno dei suoi ingressi è a 0).

In verità la soluzione, affidata in parte ad una porta logica "indifesa", rende un po' precaria l'informazione relativa ai tasti della seconda Keypad; osservando il diagramma temporale della **Figura 4** si può dedurre: a) dal momento in cui la linea *Data Available* passa a 1 il dato a 5 bit (letto su PortA) è perfettamente attendibile: negli istanti precedenti l'integrato *si è mangiato* certamente ogni rimbalzo e quelli presenti sul bit4 (in uscita dalla NAND) non essendo stati letti, sono andati perduti; b) per gran parte del tempo in cui DA rimane ad 1 la situazione è stazionaria; c) ma poco prima che torni a 0 (qualche decina di ns) il rimbalzo dovuto al rilascio del tasto *è fatto sparire* sui bit gestiti dal dispositivo ma è ancora presente sul bit4 ...

Fortunatamente la cosa è irrilevante: l'intervallo di tempo durante il quale il tasto rimane premuto è enorme (almeno 200 o 300 ms) per i tempi di una CPU che, leggendo la sua PortA sul fronte di salita di DA, non avrà alcun dubbio sulla attendibilità della codifica ricevuta in cambio da questo progetto; così come avrà la assoluta certezza del rilascio pulito del tasto aspettando il fronte di discesa di DA.

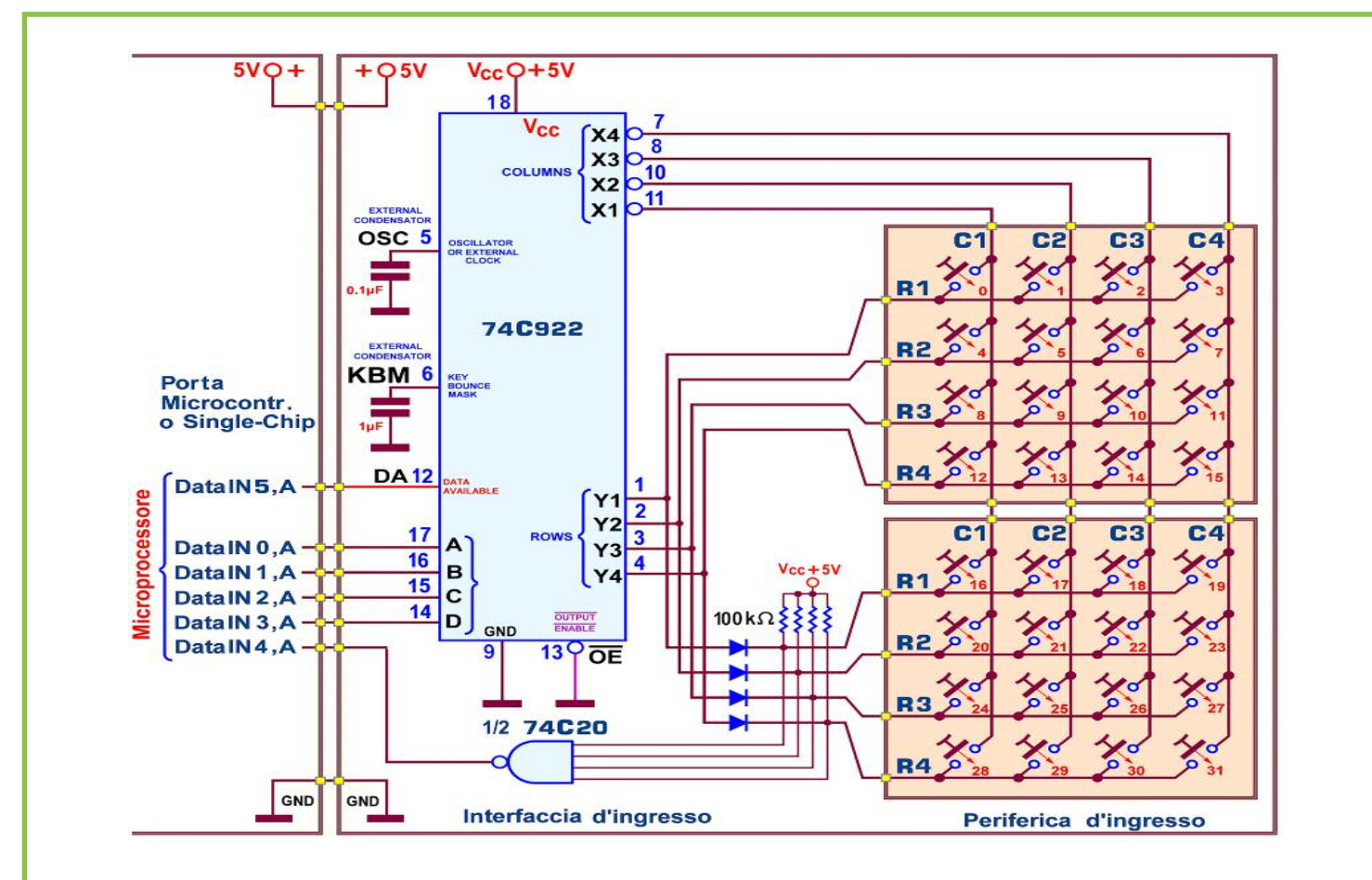



Figura 6 - Lettura di una 32 Button Keypad con Key Encoder 74C922 via PORTA di un microcontrollore

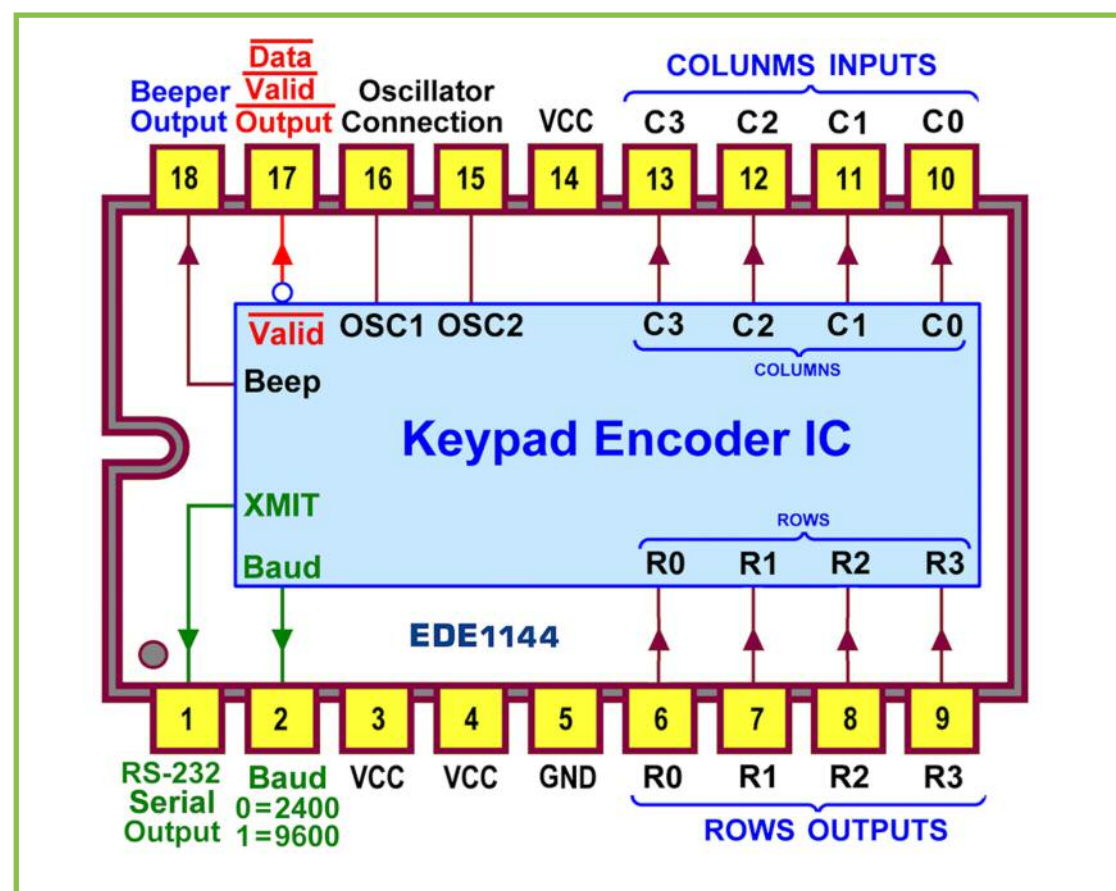



Figura 7 - 16-Key Encoder EDE1144: pinout

In chiusura parliamo di Keypad Encoder alternativi; la necessità di gestire le tastiere in modo specializzato e ottimale ha stimolato i produttori a garantire la disponibilità di numerosi circuiti integrati, più o meno costosi; a questo proposito va ricordato che la globalizzazione della rete consente la reperibilità (a prezzi interessanti) di qualunque oggetto, anche dichiarato obsoleto dai rivenditori.
Molto frequentemente sono disponibili integrati in grado di gestire anche un visualizzatore, come il CH450 (LED drive and keyboard control chip) capace di pilotare automaticamente fino a 6 Digit a sette segmenti e una Matrice di 6x8 tasti, cioè fino a 48 contatti; oppure il glorioso INTEL 8279 (keyboard and display controller) progettato specificatamente per interfacciare fino a 16 display a 7 segmenti (oppure di 128 LED) e una tastiera fino ad un massimo di 64 tasti.

Volendo prendere in considerazione un controller per Keypad alternativo al 74C922, vediamo ora i dettaglio il EDE1144 Keypad Encoder, della E-Lab Digital Engineering, progettato per interfacciare un tastierino organizzato a Matrice con 4 Righe x 4 Colonne (con 16 tasti o meno); il suo pinout è mostrato in **Figura 7**.
Anche questo dispositivo dispone della funzione avanzata di *Debouncing* dei contatti (rendendo minimo il compito del software nel microcontrollore) e, per certi versi, è più efficiente del precedente, essendo in grado di comunicare i dati relativi al tasto rilevato sia tramite le classiche 4 uscite dati parallele sia organizzandoli con protocollo RS-232 sulla sua uscita seriale, *XMIT*.
Un'altra sua caratteristica è quella di sottoporre a scansione il tastierino solo quando viene rilevata la pressione di un tasto, evitando di farlo se i segnali non cambiano; il datasheet del EDE1144, per questa sua esclusiva, lo descrive come *elettricamente silenzioso*, specificando che gli altri encoder di tastiera, eseguendo la scansione continua dei contatti, irradiano rumore EMI (ElectroMagnetic Interference, Rumori Induttivi) dai fili che portano alla tastiera, con conseguenti difficoltà (durante le prove relative alle emissioni) per la certificazione e l'approvazione finale del prodotto da parte degli enti predisposti.
La **Figura 8** mostra lo schema adatto alla lettura dei 16 contatti di un tastierino 4x4 e ci aiuta a comprendere i dettagli funzionali; per evitare cortocircuiti durante il ciclo di scansione (nel caso in cui venga premuto più di un tasto alla volta) le Righe da R0 a R3 sono collegate a quelle della Keypad attraverso resistori da 330 Ohm; la loro presenza può contribuire anche a ridurre le emissioni elettromagnetiche generate dai fili della tastiera.
Per evitare di lasciarli fluttuanti (cioè scollegati, quindi soggetti a potenziali interferenze logiche) su ciascuno dei 4 ingressi di Colonna (da C0 a C3) deve essere presente un resistore di pulldown di 4,7KOhm, anche se si dovesse utilizzare una tastiera con un numero di tasto minore di quello massimo, 4x4; su OSC1 (pin16) e OSC2 (pin15) può essere collegato (come in figura) un quarzo da 4MHz, ma, volendo utilizzare un clock TTL esterno, esso andrà applicato su OSC1, lasciando OSC2 scollegato.
Le Righe sono tenute a 1 (tutte alte), in attesa della pressione di un tasto, e non appena ciò avviene saranno commutate a 0 in sequenza per sottoporre a tastiera a scansione; in questo modo il EDE1144 è in grado di determinare quale tasto è stato premuto e di organizzare l'informazione per il microcontrollore: a) sulle uscite da R0 a R3 viene posto il nibble corrispondente al valore binario del numero d'ordine del tasto premuto (da 0000 per il tasto_0 a 1111 per il tasto_15); queste stesse uscite sono rese disponibili sulle linee (programmate in ingresso) di una Porta del micro (per esempio PortA) per assicurare la lettura parallela del dato; b) un microsecondo dopo viene attivata (bassa) l'uscita *Data Valid* (pin17): essa rimarrà a 0 per 50 ms, durante i quali l'informazione sui 4 pin di uscita verrà congelata e mantenuta stabile, anche in presenza di eventuali ulteriori chiusure (rimbalzi) di quel contatto o della pressione di qualsiasi altro contatto, c) contemporaneamente, sull'uscita *Beeper Output* (pin18) viene generata un'onda quadra di 1 kHz, per 45 ms; con un piccolo amplificatore a transistor ad emettitore comune (un qualunque NPN, come il 2N2222 o il BC107, vedi **Figura 8**) è possibile pilotare un cicalino piezoelettrico o un piccolo altoparlante con cui fornire anche una segnalazione acustica, per avvertire l'operatore di tastiera che è stato premuto un tasto; d) non appena il beeper termina di suonare, negli ultimi 5 ms, sulla linea seriale *XMIT* (pin1, rimasta alta fino a questo istante) vengono trasmessi i dati relativi alla pressione del tasto; questa operazione consuma un tempo relativamente breve, per cui, al termine, viene attivato un ritardo di durata variabile (in funzione della velocità di trasmissione selezionata) per far si che l'intero processo possa durare esattamente 50 ms.
In **Figura 9** il processo, appena descritto, è

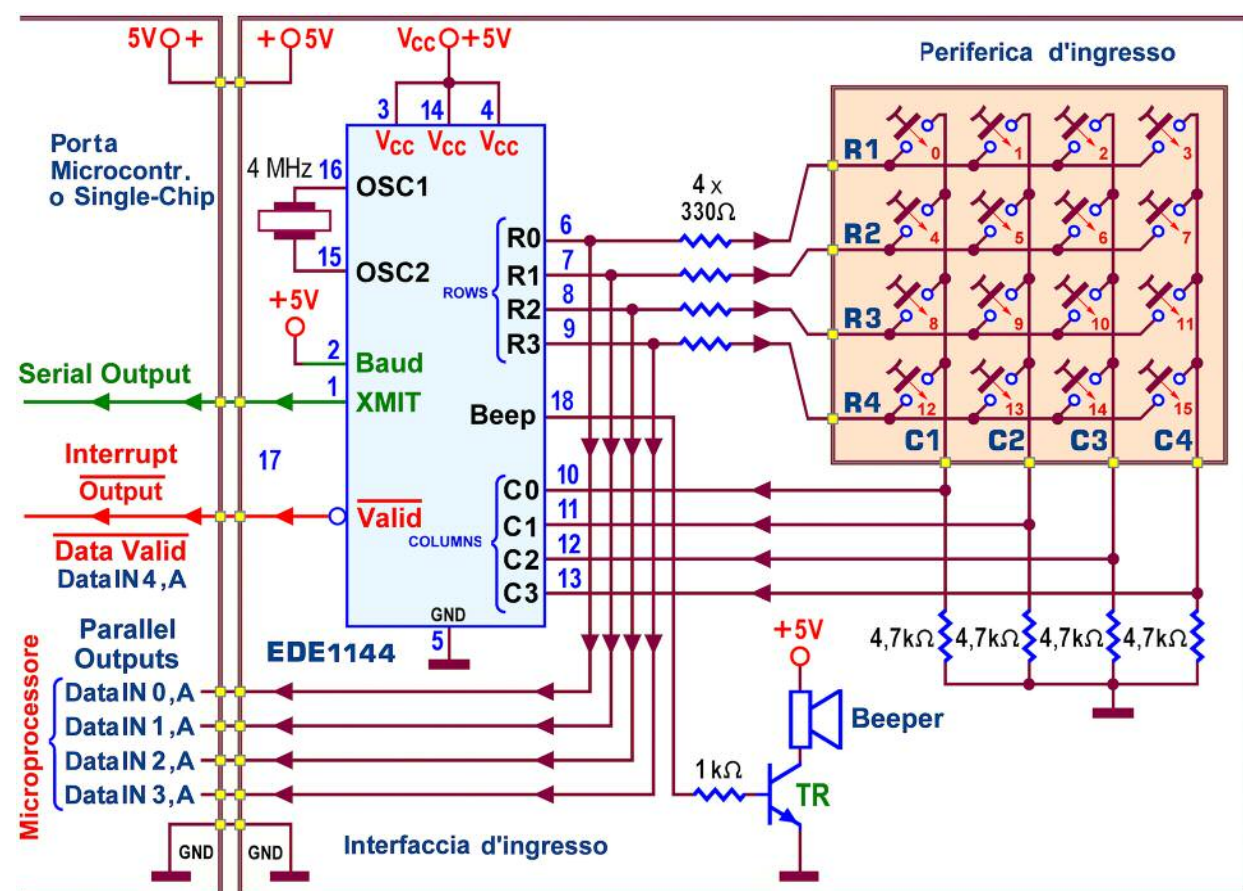



Figura 8 - Lettura di una 16 Button Keypad con Key Encoder EDE1144 via microcontrollore

tradotto nel diagramma temporale corrispondente; una delle caratteristiche del EDE1144 è la funzione *Key Auto-Repeat*: se un tasto viene tenuto più a lungo di 660 ms (ritardo di autorepeat) le uscite vengono cancellate e l'intero processo viene ripetuto 5 volte al secondo (esattamente ogni 204 ms) finché il tasto viene rilasciato.
In conclusione, la pressione di un tasto (e l'eventuale sua ripetizione automatica) può essere gestita in due modi, alternativi tra loro, utilizzano le 4 uscite parallele o l'unica uscita seriale; in entrambi è utile affidarci al segnale *Data Valid*, disponibile non appena un tasto è premuto e dopo ogni ciclo di ripetizione (se esso viene mantenuto premuto).
Nella gestione parallela, il software ha ben 50 ms per interrogare in polling questo segnale e per leggere le uscite dati (da R3 a R0, bloccate per tutto il tempo sul valore appropriato) non appena esso viene attivato (basso) dal processo di acquisizione; ma può essere conveniente utilizzare DA per controllate la linea di interrupt del processore, evitando il tempo perso in attesa e consentendo di impiegarlo per altre attività. Ovviamente l'uscita seriale potrà, in questo caso, essere ignorata.
Nella gestione seriale, l'uscita *XMIT* (pin1) trasmette dati seriali al microcontrollore, alla velocità di trasmissione selezionata tramite l'ingresso *Baud Rate* (pin2, 2400 baud con 0 e 9600 baud con 1); la sequenza degli 11 bit è quella prevista dal tradizionale formato RS-232 (1,8,N,1, un bit di Start, 8 bit di Dato, un bit di Stop, Nessuna Parità); da notare che il dato seriale trasmesso non è più il numero d'ordine del tasto premuto ma il byte che esprime il carattere ascii del simbolo del sistema di numerazione esadecimale ad esso corrispondente: '0'=30H=00110000 per 0000 (tasto_0), ..., '9'=39H=00111001 per 1001 (tasto_9), 'A'=41H=01000001 per 1010 (tasto_10), ..., 'F'=46H=01000110 per 1111 (tasto_15).
Il segnale disponibile sulla linea XMIT viene offerto con livelli di tensione TTL/CMOS, cioè tra 0V e 5V e, di solito, può essere trasferito direttamente all'UART del microcontrollore così com'è; tuttavia, se i dati devono essere inviati ad un dispositivo che si aspetta livelli di tensione RS-232 (tra -12V e +12V, come la porta seriale di un PC), sarà necessario inserire un integrato di conversione di livello di tensione (come il MAX232) tra il pin1 di uscita seriale del EDE1144 e pin RxD di ingresso seriale del PC.
La scelta seriale può essere gestita in due modi diversi; la prima opzione permette di ridurre da otto a una le linee di I/O necessarie per leggere la Keypad ed è applicabile se il microcontrollore dispone di un UART "a bordo"; in questo caso, per monitorare i dati seriali appena ricevuti, basta interrogare il bit *Received Data Ready* (o *Data Available*) del Registro di Stato Linea dell'UART del microcontrollore; alternativamente il micro può essere configurato per generare un interrupt quando un nuovo byte è ricevuto dal UART.
Il secondo metodo è l'unico possibile se il microcontrollore non dispone di UART: come per la gestione parallela è necessario interrogare il segnale *Data Valid*; poiché esso è attivato ben prima che siano trasmessi i dati seriali, il programma di gestione ha a disposizione una pausa di 45ms (durante la quale viene generato il segnale *Key Beep*) prima che venga inviato il bit di Start e i bit rimanenti, durante la quale può finire di fare quello che sta facendo, mantenendo attiva la procedura di ricezione seriale fino a quando viene rilevato il bit di Start del EDE1144.

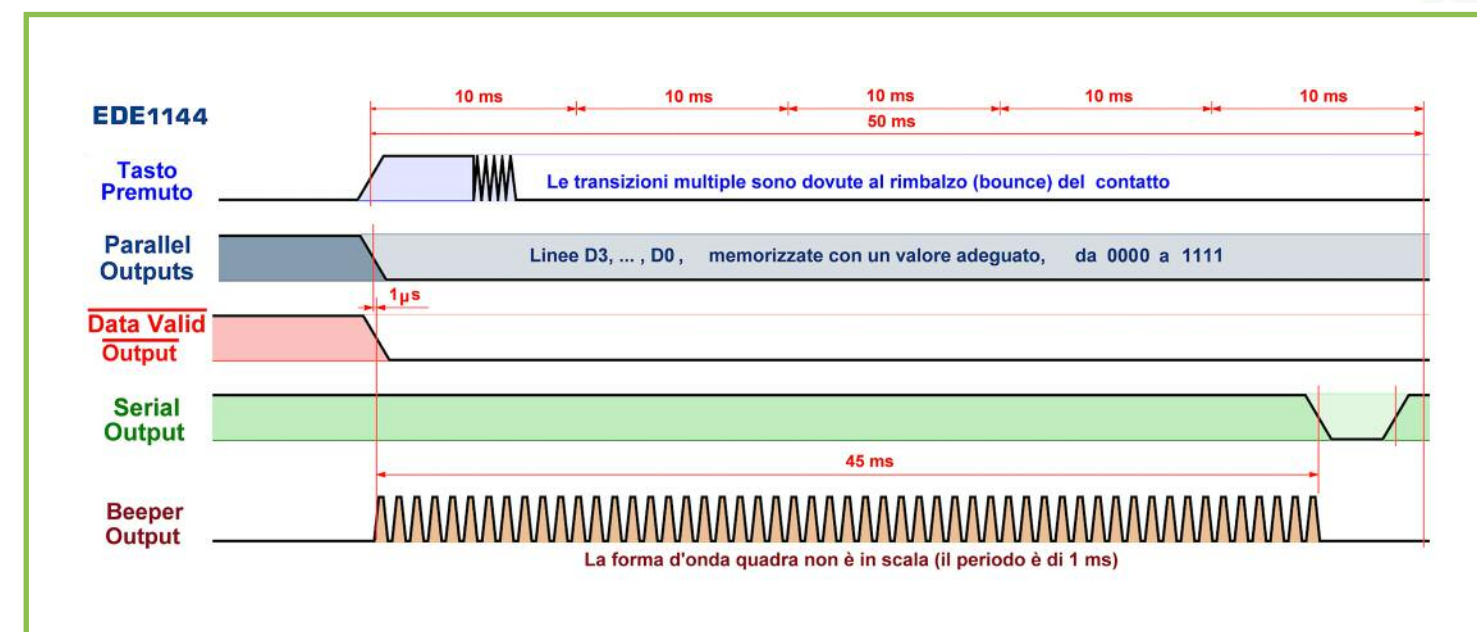



Figura 9 - 16-Key Encoder EDE1144: diagramma temporale

di DANIELE CAPPA

COSTRUIRE UN WATCH WINDER

Dietro al nome anglosassone si nasconde una scatola carica orologi

Su richiesta di un amico ho messo insieme l'elettronica di comando di una scatola carica orologi…
Molti, come me fino a qualche giorno fa, si chiederanno cosa diamine sia una scatola carica orologi? Si tratta di un contenitore provvisto di alcuni piatti, o dei "polsi finti", su cui è possibile fissare orologi da polso. L'utilizzo è ad esclusivo appannaggio degli esemplari meccanici con carica automatica, quelli che si caricano portandoli semplicemente al polso. E' evidente che al polso possiamo indossare un orologio per volta, se ne abbiamo molti abbiamo due soluzioni possibili: indossarli a rotazione, oppure una scatola carica orologi.
Meccanicamente si tratta di uno o più motori elettrici, provvisti di riduttore in modo che girino pianopiano, su cui è montato un piattello su cui andrà fissato/appoggiato l'orologio da caricare. Il piattello gira in un senso per qualche minuto, si ferma e dopo qualche minuto riparte invertendo il senso di rotazione. Il ciclo si ripete all'infinito.
Oggetti di questo tipo che possano contenere alcuni orologi costano cifre importanti, dunque ecco la nostra realizzazione.

Foto 1 - Il prototipo

L'IDEA DELLA MECCANICA

Sono stati utilizzati alcuni motorini a 12V, ad uso automobilistico. Si tratta dei servi che aprono e chiudono le bocchette del riscaldamento/condizionatore all'interno dell'auto.
Piccoli, silenziosi, poco avidi di corrente (funzionano già con soli 30 mA a 12V e a

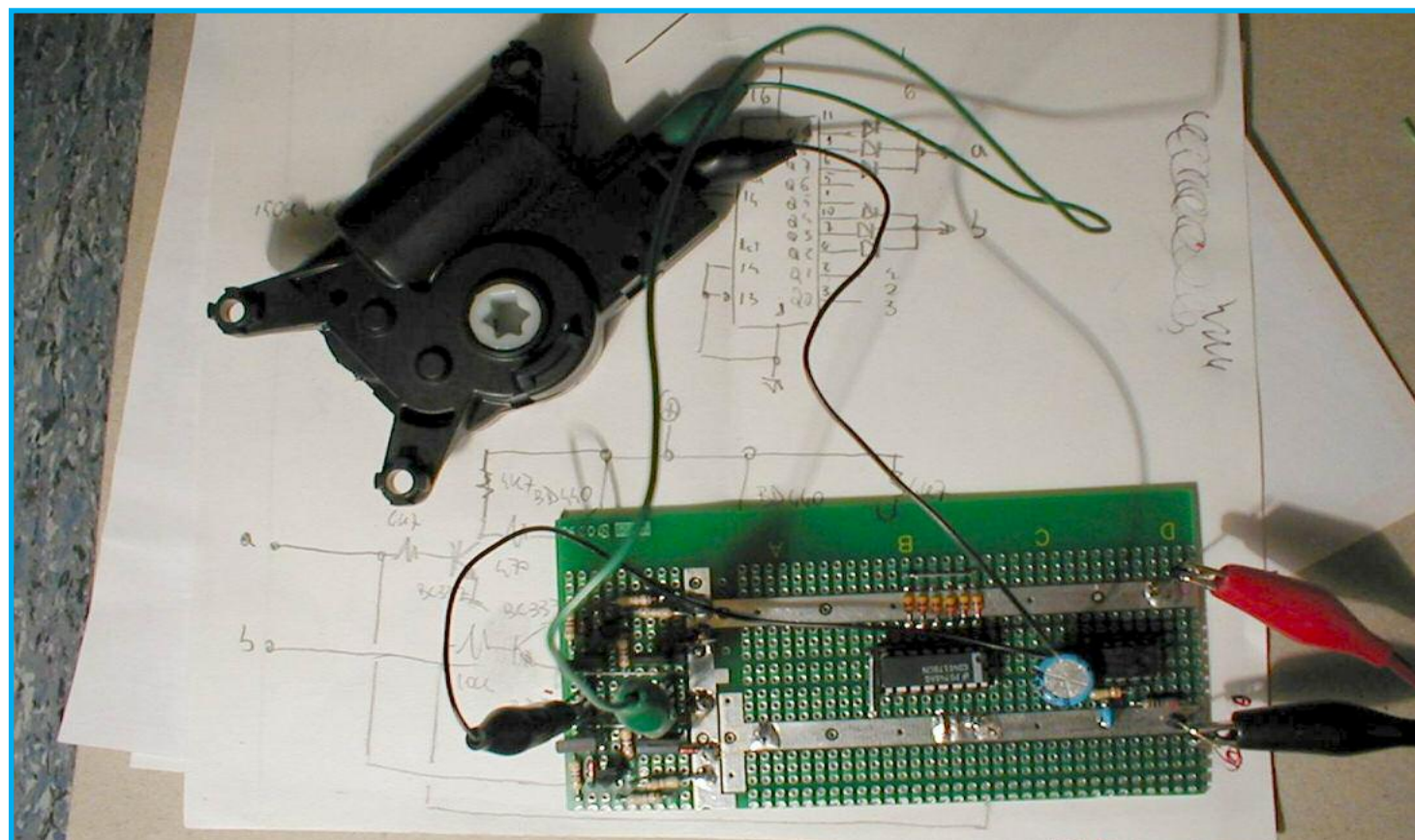
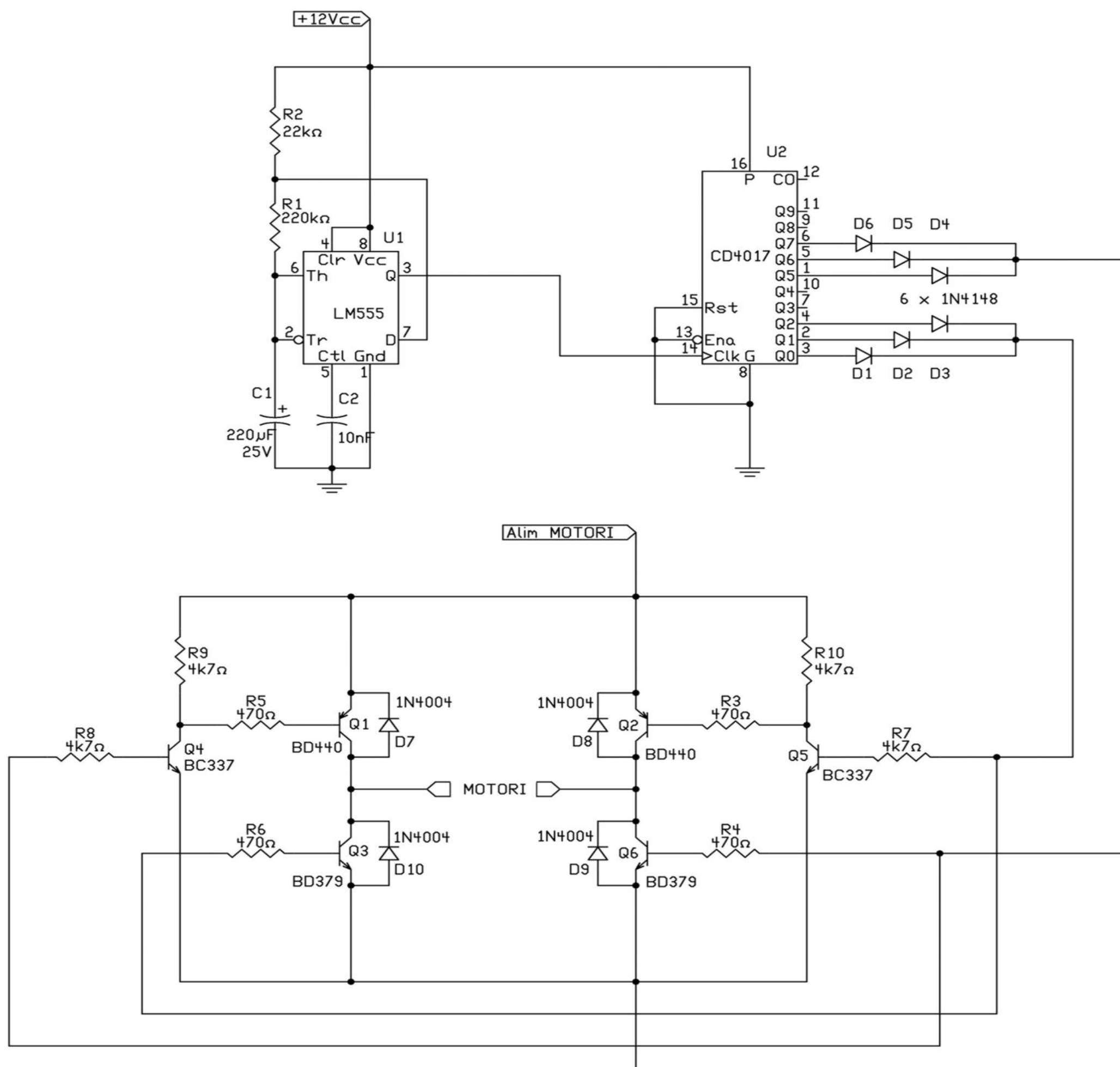



Figura 1 - Schema elettrico Watch winder

5V è praticamente impossibile fermare l'uscita con le dita), girano pianissimo… merito delle quattro riduzioni meccaniche, la prima a vite senza fine e le successive a coppie di ruote dentate. Hanno un problema di fondo, come ricambio costano cifre importanti (130 euro/cad per gli esemplari Audi/VW visibili nelle foto), ma una visita da un autodemolitore, o una scelta oculata su modelli dalle quotazioni più modeste risolve il problema.
Alcuni piatti inseriti sulla boccola di uscita del riduttore completano la meccanica "mobile", il resto non è che una scatola…

L'ELETTRONICA

Un generatore di clock con tempi piuttosto lunghi costituito dal solito LM555 in configurazione astabile che abbiamo già visto in tutte le salse, cui fa seguito un vil contatore decimale, un CD4017 sulle cui uscite sono prelevati i segnali che comanderanno un ponte ad H costituito da quattro transistor e due inverter, sempre a transistor.
L'intenzione originale era di utilizzare quattro darlington, ma il cassettino offriva solamente esemplari il cui guadagno è modesto… Il 4017 non fornisce molta corrente in uscita, ma i motori utilizzati sono davvero molto parchi nel consumo, dunque ho impiegato quanto disponibile ottenendo già un largo margine di sicurezza sul funzionamento… anche se le modifiche su millefiori sono sempre fastidiose, ritengo di non doverli sostituire. Vediamo dunque la logica di funzionamento…
Per i primi tre impulsi di clock sono attive le prime tre uscite del 4017 (Q0, Q1 e Q2) con cui il motore gira in un senso, poi si ferma per due impulsi di clock (ovvero quando so-

Figura 2: una scatola commerciale

no attive le successive due uscite del 4017, Q3 e Q4, che non sono collegate a nulla) dopo di che riparte in senso contrario per altri tre impulsi…in coincidenza con il passaggio a livello alto di Q5, Q6 e Q7 del contatore. Poi sta fermo per due impulsi (Q8 e Q9)… poi riparte per tre da Q0 e seguenti… così all'infinito.

I diodi sulle uscite utilizzate formano una porta OR discreta che comanda direttamente la basi dei due finali NPN, oppure le basi dei due inverter a transistor che si occupano a loro volta di comandare i due finali PNP.

I due sensi sono ottenuti utilizzando un ramo o l'altro del ponte ad H, dunque eliminando un diodo si abbassa il tempo di rotazione, aggiungendolo si aumenta, sempre del tempo corrispondente ad un impulso di clock che con i valori proposti è piuttosto lungo, 75 secondi circa.

MONTAGGIO

Il tutto è stato assemblato su un ritaglio di millefiori da prototipi, di quelle che hanno delle piste già presenti e che si utilizzano per prototipi in cui si impiegano chip DIL, per questo niente circuito stampato.

In un paio di sere lo si mette insieme senza problemi, non sono necessarie tarature e funziona al primo colpo.

I componenti vanno dimensionati secondo l'assorbimento dei motori impiegati, in questo caso una coppia NPN/PNP da 5-600 mA sarebbe stata più che sufficiente. La scelta dei BD440 e BD379 è dettata esclusivamente dalla disponibilità del momento.

Una coppia di darlington stile TIP140/141/142 e TIP145/146/147 assolvono alla funzione permettendo un carico importante (fino ad alcuni A) senza altre modifiche se non un aumento della resistenza di base fino a 4700 - 8200 ohm. Se i motori sono più grandi sostituirei anche i diodi posti tra collettore ed emettitore dei transistor con qualcosa di più veloce in grado di sopportare un paio di centinaia di volt. Nel caso un piccolo radiatore su cui montare i finali potrebbe essere una buona idea. Nel caso è necessario impiegare il kit di isolamento, pena ritrovarsi con tutti i collettori collegati insieme e conseguente corto dell'alimentazione. La logica rimane comunque invariata.

Il contatore decimale è sostituibile con un CD4022, contatore ottale, rivedendo solamente le uscite (ne ha ovviamente due in meno), dunque è utilizzabile prevedendo le uscite Q0 e Q1 con motore acceso in un senso, Q2 e Q3 motore spento (ovvero senza diodi sull'uscita), poi Q4 e Q5 per l'altro senso di rotazione per finire con Q6 e Q7 nuovamente motore spento, il ciclo si ripete anche qui all'infinito.

I due chip hanno una piedinatura molto simile e l'utilizzo di uno o dell'altro non comporta praticamente alcuna differenza, se non adeguare le uscite secondo necessità.

Il clock generato dal 555 è cosa nota, nulla da dire in merito… aumentando, o diminuendo il valore di C1 (o di R1) , aumenta o diminuisce il periodo di clock e con esso i tempi di funzionamento e pausa dei motori. Unica raccomandazione, R1 deve avere un valore molto più grande di quello di R2.

All'accensione il contatore non è resettato e C1 è scarico, dunque il primo ciclo potrebbe avere tempi diversi da quanto ci si aspetta, dato l'uso a cui è destinato il tutto questo non rappresenta certo un problema.

L'alimentazione è stata ricavata da un ex_caricabatterie di chissàcosa… 9V per alcune centinaia di mA con cui funziona perfettamente con quattro gemelli del motore visibile nella foto. La logica funziona da 3 a 18V, secondo lo standard delle porte Cmos della serie 4000, Il consumo del comando è molto basso, pochi mA sono in grado di far funzionare la parte logica. Il consumo sarà quasi esclusivamente a carico dei motori, dunque l'alimentatore andrà dimensionato in base a questi. L'alimentazione della logica e dei motori può essere diversa, ma sempre in corrente continua e con la massa in comune.

I motori devono poter girare in un senso o nell'altro capovolgendo l'alimentazione, dunque niente motorini di ex_ventoline da PC!! Dato l'uso casalingo è fondamentale che questi siano silenziosi, molto silenziosi.

LA SCATOLA

Il contenitore in questo caso ricopre un ruolo importante nella realizzazione del tutto, si tratta di un oggetto che va tenuto in vista, dunque un bello scatolino di legno, meglio se provvisto di coperchio, magari in plexiglass… La parte meccanica – estetica della realizzazione non era tra i miei compiti, dunque non sono in grado di farvela vedere. Qui ognuno si dovrà necessariamente arrangiare al fine di ottenere un risultato all'altezza del perfetto bricoleur.

Ricordandosi sempre che i lavori vengono male da soli, non è necessario farli male! (nota di Gianni).

di FRANCESCO FICILI

MOUSE USB CON PIC18F2550

In questo articolo verrà illustrata la realizzazione di un mouse USB, basato sul microcontrollore RISC PIC18F2550 (dotato di controller USB slave integrato) e sullo stack USB open source della Microchip. Il circuito presentato implementa la classe USB HID, secondo le specifiche 2.0

Quando si inizia a fare uso e pratica con i microcontrollori, le primissime idee di progetto riguardano oggetti di uso comune, data la tendenza del progettista a realizzare qualcosa di utile con le proprie mani, come ad esempio orologi digitali, termometri digitali, lettori mp3, etc. Il progetto proposto in questo articolo vuole essere d'aiuto a chi avverte quest'esigenza ma ha bisogno della classica "spintina" per iniziare. Descriveremo infatti come realizzare un semplice mouse USB, utilizzando un RISC ad 8-bit e pochissimi altri componenti.

Data la natura dimostrativa del progetto non verrà realizzato un vero e proprio mouse ottico, ma faremo uso di quattro tasti per impartire la direzione del puntatore sui due assi e di altri due pulsanti per emulare i tasti destro e sinistro dei tipici mouse da desktop.

A parte questo aspetto il risultato del progetto sarà una vera e propria periferica USB, utilizzabile su qualsiasi PC, naturalmente purchè dotato di almeno una porta USB 2.0. Naturalmente, anche per un progetto che si presenta come un puro "esercizio di estetica" non mancano le possibili applicazioni reali; nel caso in esame è possibile immaginare di utilizzare un oggetto simile in ambienti industriali, dove la presenza di polvere o l'assenza di un piano di appoggio rendono impossibile l'uso di un classico mouse da PC. Possibili evoluzioni di quest'idea possono essere la sostituzione dei tasti direzionali con un mini-joystick o con una trackball.

SCHEMA ELETTRICO

L'immagine di figura 1 mostra lo schema elettrico del dispositivo. Come si vede, il cuore del circuito è costituito dal microcontrollore ad 8-bit PIC18F2550, prodotto da Microchip Technology. Il PIC18F2550 è uno dei PIC18 direttamente interfacciabili alle porte USB dei comuni desktop, senza l'uso di controller o transceiver esterni. Infatti questo piccolo micro dispone al suo interno di tutte le risorse hardware necessarie all'implementazione di una periferica USB slave.

Il PIC18F2550 è cloccato con un quarzo da 4 MHz, collegato ai piedini OSC1 ed OSC2. Tramite questa sorgente di clock primaria viene ottenuto, attraverso un PLL interno e un divisore di frequenza, una sorgente a 48 MHz che temporizza l'intero sistema. I 48 MHz di clock sono indispensabili per il corretto funzionamento dell'USB FS (Full Speed), ecco perché i PIC della famiglia 4550/2550 dispongono, all'interno della loro architettura hardware, del PLL. Oltre al quarzo, viene impiegato il condensatore C3, da 330 nF, necessario alla stabilizzazione del regolatore interno a 3,3V del PIC. Anche questo regolatore è funzionale alla periferica USB, essendo la logica di quest'ultima a 3,3V, mentre quella del resto delle periferiche del PIC è a 5V. E' presente ancora il connettore J1, un connettore USB tipo B femmina, da stampato.

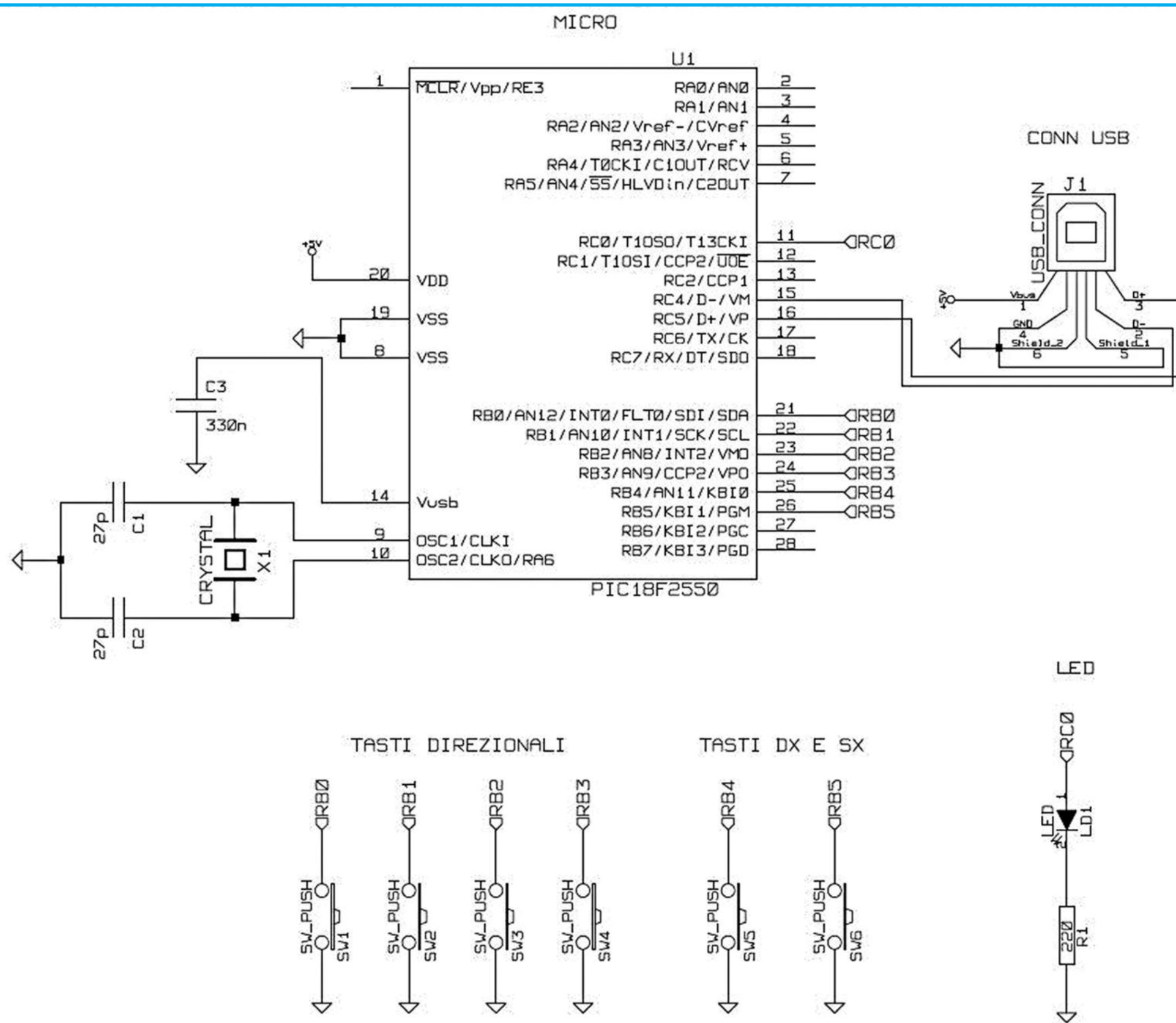



Figura 1 Schema elettrico del mouse USB

Non è proibito l'uso di un altro connettore, ma lo standard indica in questo il connettore da utilizzare per l'implementazione di periferiche USB slave. Lo schema prevede ancora 6 pulsanti normalmente aperti. In generale è possibile utilizzare qualsiasi tipo di switch normalmente aperto. Nel nostro caso sono stati utilizzati 6 pulsanti miniatura d stampato. I primi 4 (SW1-SW4) costituiscono il blocco di tasti direzionali (in sostanza emulano la sfera o il LED degli ormai più comuni mouse ottici), mentre SW5 ed SW6 sono il tasto destro e sinistro del nostro mouse. I 6 switch sono stati collegati alla porta B (ingressi RB0-RB5). La scelta non è casuale, in quanto questa è l'unica porta del PIC ad essere dotata di un blocco di resistenze di pull-up, attivabili via software. Tale scelta permette di risparmiare sei resistenze. Ultimo (ma non per importanza) il LED LD1, con la sua resistenza di limitazione R1, da 220 Ohm. Questo LED servirà da indicatore luminoso per evidenziare lo stato di funzionamento della periferica, che nel caso delle periferiche USB in generale, e di quelle autocostruite in particolare, costituisce un'importante informazione di debug.

L'uso di così pochi componenti per la realizzazione di una periferica di questo tipo è reso possibile dall'impiego del PIC18F2550, che elimina la necessità di un transceiver USB esterno, dato che all'interno della sua architettura ingloba una SIE (Serial Interface Engine), visibile in figura 2. Come si vede, all'interno del micro sono disponibili tutte le risorse hardware per la realizzazione di una periferica USB. Oltre al transceiver, è presente il regolatore a 3,3V, 1 Kbyte di dual port RAM dedicata, 2 resistenze di pull-up sulle linee D+ e D- (per la selezione delle modalità Full Speed e Low Speed), oltre alla logica della SIE (blocco USB SIE). Questo complesso insieme di periferiche hardware permette di limitare la componentistica esterna ad un condensatore (per il regolatore a 3,3V) ed al connettore USB.

FIRMWARE

Per la realizzazione del firmware è stato utilizzato lo stack USB open source di Microchip, ed in particolare la versione 2.4. La tabella 1 mostra, in maniera riassuntiva, l'evoluzione dello stack, che dal 2005

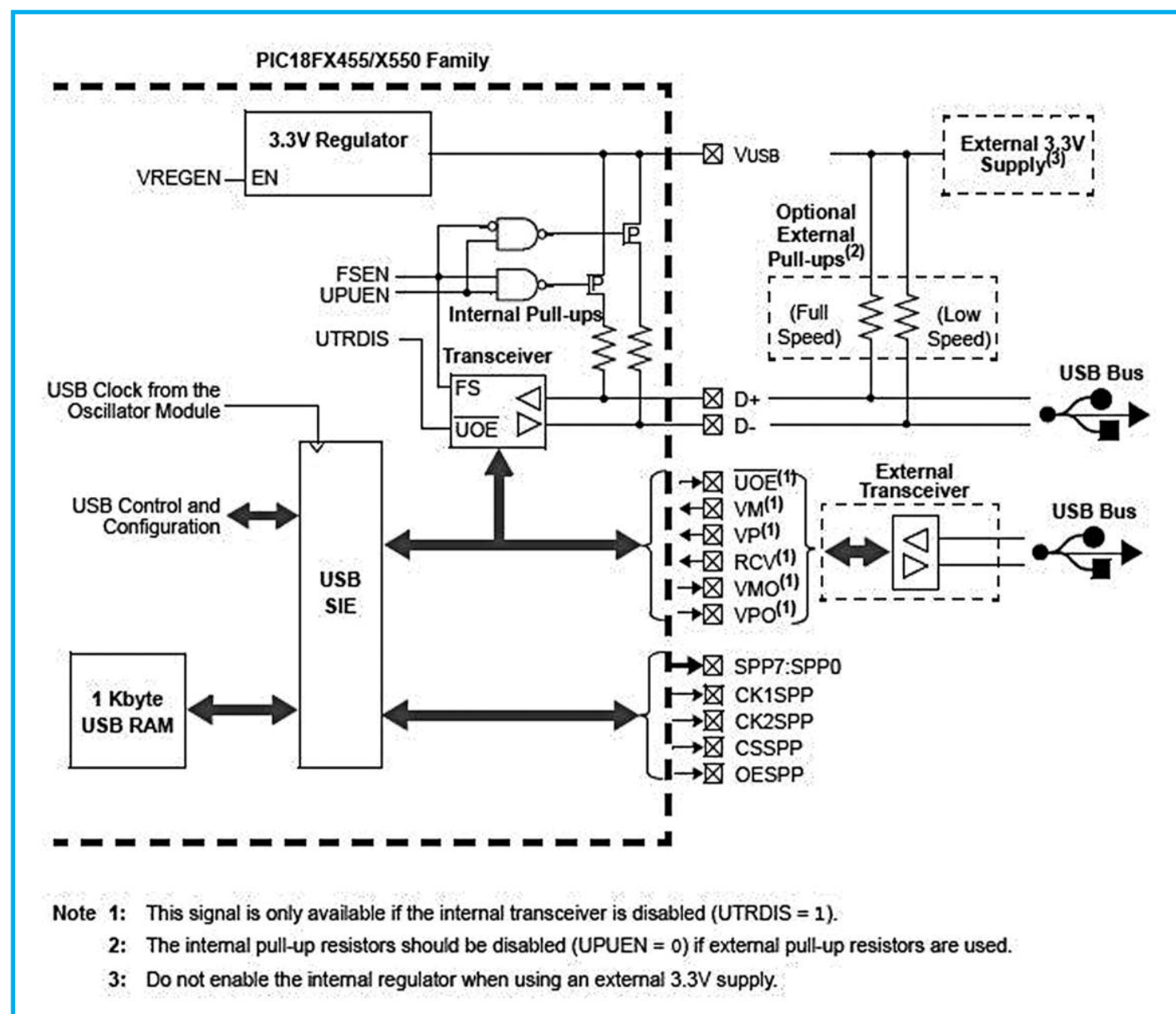



Figura 2 SIE del PIC18F2550

circa ad oggi ha subito notevoli modifiche ed aggiornamenti. Maggiori informazioni sullo stack possono essere reperite sul sito web di Microchip Technology (www.microchip.com), all'interno della sezione USB connectivity.

Per la realizzazione del Firmware è stato utilizzato l'ambiente di sviluppo Microchip MPLab IDE 8.10, con il compilatore C18 Student edition. La classe utilizzata è naturalmente la classe USB HID (Human Interface Device – Dispositivi di interfaccia uomo-macchina), che comprende le definizioni e le specifiche adottate dall'USB Implementer Forum's Device Working Group per periferiche come tastiere, joystick, joypad e naturalmente mouse . In particolare per periferiche come mouse e tastiere, lo standard USB ha progressivamente rimpiazzato il collegamento PS/2, tanto che ormai nei PC di ultima generazione questa tipologia di porte tende ormai a scomparire.

In figura 4 è riportata la schermata del project manager di MPLab IDE per questo progetto.

Versione Stack	Classi implementate	Note
1.0	HID (mouse), CDC (generica), MSD (generica)	Prima release
1.3	HID (mouse), CDC (generica), MSD (generica), Custom class (mchpusb.dll, bootloader)	Nessuna
2.3	HID (keyboard, mouse, joystick, bootloader), CDC (generica, serial emulation), MSD (internal flash, SD reader)	Integrazione del Graphic stack
2.4	HID (keyboard, mouse, joystick, bootloader), CDC (generica, serial emulation), MSD (internal flash, SD reader), Custom Class (LibUSB, Bootloader), USB Charger	Supporto per la famiglia PIC18F46J50.

Tabella 1: Evoluzione dello stack USB open source di Microchip

Figura 3 Roadmap microcontrollori Microchip con periferica USB slvae/host integrata

Lo stack implementa tutte le specifiche necessarie affinché, una volta collegata la periferica al PC, quest'ultimo ne riconosca la tipologia, la inserisca tra le periferiche attive, allochi una frazione di banda sul bus per garantire la comunicazione e predisponga i buffer di input/output per lo scambio dei dati. Una descrizione esaustiva di tutte le funzioni che compongono il firmware con cui è stato programmato il PIC18F2550 esula dogli scopi di quest'articolo, d'altra parte non basterebbe l'inte-

LISTATO 1

Definizioni riportate su HardwareProfile.h

Download

```
/** SWITCH ***************************************************/
    #define    mInitAllSwitches()
     TRISBbits.TRISB0=1;TRISBbits.TRISB1=1;TRISBbits.TRISB2=1;
     TRISBbits.TRISB3=1;TRISBbits.TRISB4=1;TRISBbits.TRISB5=1;
    #define mInitSwitch2()      TRISBbits.TRISB0=1;
    #define mInitSwitch3()      TRISBbits.TRISB1=1;
    #define mInitSwitch4()      TRISBbits.TRISB2=1;
    #define mInitSwitch5()      TRISBbits.TRISB3=1;
    #define mInitSwitch6()      TRISBbits.TRISB4=1;
    #define mInitSwitch7()      TRISBbits.TRISB5=1;

    #define sw2                 PORTBbits.RB0
    #define sw3                 PORTBbits.RB1
    #define sw4                 PORTBbits.RB2
    #define sw5                 PORTBbits.RB3
    #define sw6                 PORTBbits.RB4
    #define sw7                 PORTBbits.RB5
```

LISTATO 2

Funzione ProcessIO()

Download

```
void ProcessIO(void)
{
    //Blink the LEDs according to the USB device status
    BlinkUSBStatus();

    // User Application USB tasks
    if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;

    //Call the function that emulates the mouse
    Emulate_Mouse();

}//end ProcessIO
```

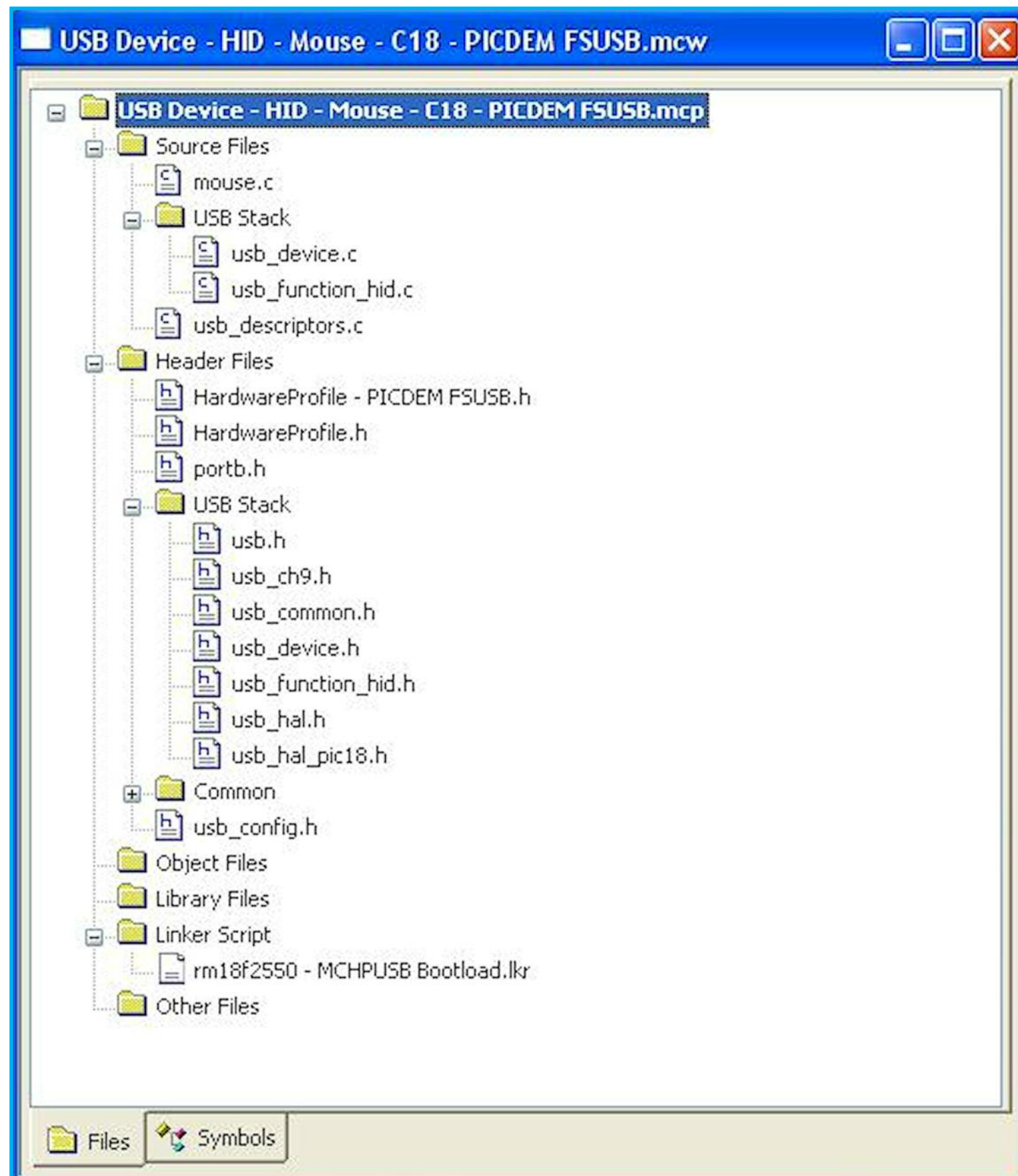



Figura 4 Project Manager del progetto Mouse USB

ra rivista, ci concentreremo quindi sulla descrizione delle funzioni più importanti.
Per prima cosa, vediamo come sono stati modificati gli hardware profiles dello stack (originariamente pensato per la scheda di sviluppo PICDEM FS USB). In pratica sono state definite sei costanti che corrispondono ai sei switch fisicamente collegati alla porta B. Il codice riportato all'interno del listato 1 illustra quando detto.

Passiamo adesso ad alcune funzioni contenute all'interno del file mouse.c, analizzando quelle più caratteristiche. Oltre alle varie routine di configurazione ed inizializzazione delle varie periferiche del PIC (che in questa sede vengono saltate), questo file contiene alcune funzioni fondamentali, come la funzione ProcessIO(). Questa funzione, il cui codice è riportato all'interno del listato 2, è una sorta di place holder per i

LISTATO 3
Funzione Emulate_Mouse()

Download

```
void Emulate_Mouse(void)
{
          if(sw2==0) buffer[2] = -1;
          else if(sw3==0) buffer[2] = 1;
          else buffer[2] = 0;

          if(sw4==0) buffer[1] = 1;
          else if(sw5==0) buffer[1] = -1;
          else buffer[1] = 0;

          if(sw6==0) buffer[0] = 1;
          else if(sw7==0) buffer[0] = 2;
          else buffer[0] = 0;

    if(HIDTxHandleBusy(lastTransmission) == 0)
    {
        //copy over the data to the HID buffer
        hid_report_in[0] = buffer[0];
        hid_report_in[1] = buffer[1];
        hid_report_in[2] = buffer[2];

        //Send the 3 byte packet over USB to the host.
        lastTransmission = HIDTxPacket(HID_EP, (BYTE*)hid_report_in, 0x03);

    }
}//end Emulate_Mouse
```

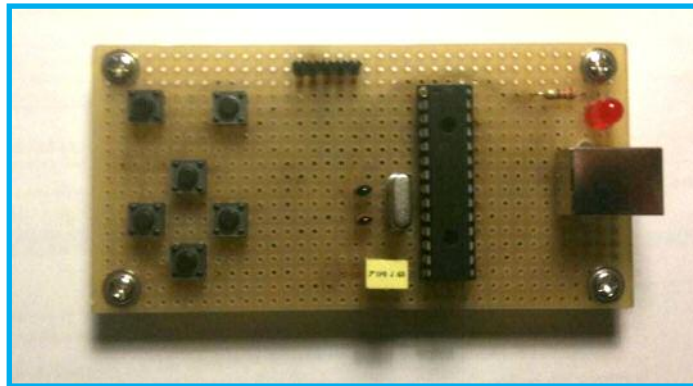



Figura 5 Prototipo del mouse USB

vari task che vengono eseguiti all'interno del ciclo principale del firmware, ed è costituita da un misto di task USB e non-USB. E' una delle poche funzioni ad essere stata mantenuta sostanzialmente invariata durante tutte le evoluzioni dello stack. La funzione ProcessIO() fa fondamentalmente tre cose: ad ogni ciclo di esecuzione richiama la funzione BlinkUSBStatus() che gestisce il LED collegato ad RC0, dopo controlla (tramite una istruzione if) se la periferica si trova in uno stato inferiore a quello di dispositivo configurato o è stata sospesa, ed in caso affermativo esce. Questo controllo serve ad evitare l'esecuzione della funzione successiva nel caso la periferica non sia attiva (e cioè nel caso in cui si trovi nello stato SUSPENDED o non abbia ancora terminato il processo di enumerazione). Infine esegue la funzione Emulate_Mouse(), che contiene il codice C che effettivamente acquisisce lo stato dei sei pulsanti collegati sulla porta B ed invia i dati sul bus USB. Analizziamo quindi proprio la funzione Emulate_Mouse(), in modo da comprendere come vengono acquisiti ed inviati i dati sul bus USB. Una periferica USB come un mouse, può venire gestita, dal punto di vista del protocollo, con un buffer di 3 byte, nel quale il primo byte indica lo stato dei pulsanti, e i due successivi indicano velocità e senso di movimento del puntatore nei due assi. Qundi, in

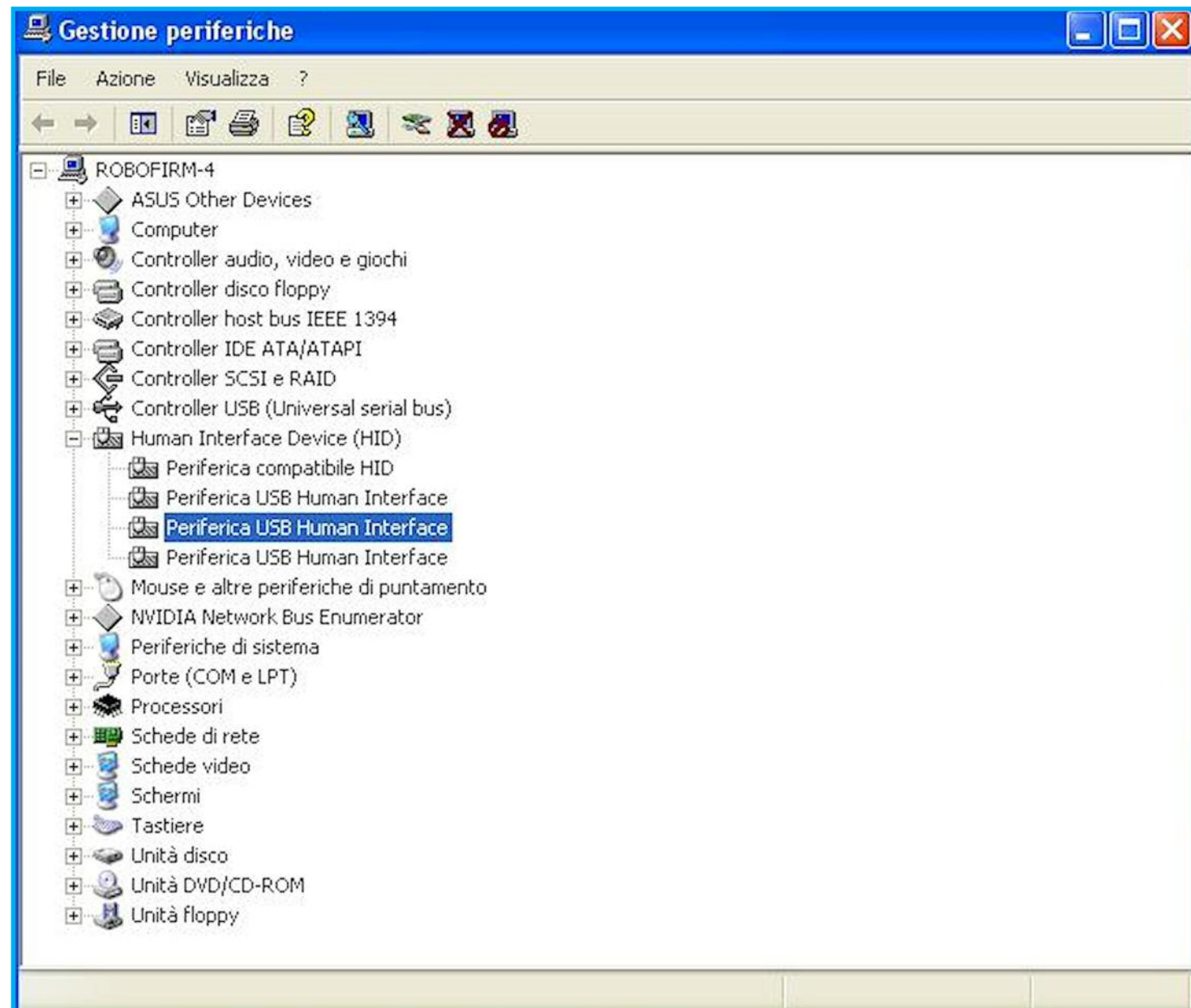



Figura 6 Risorse hardware di sistema

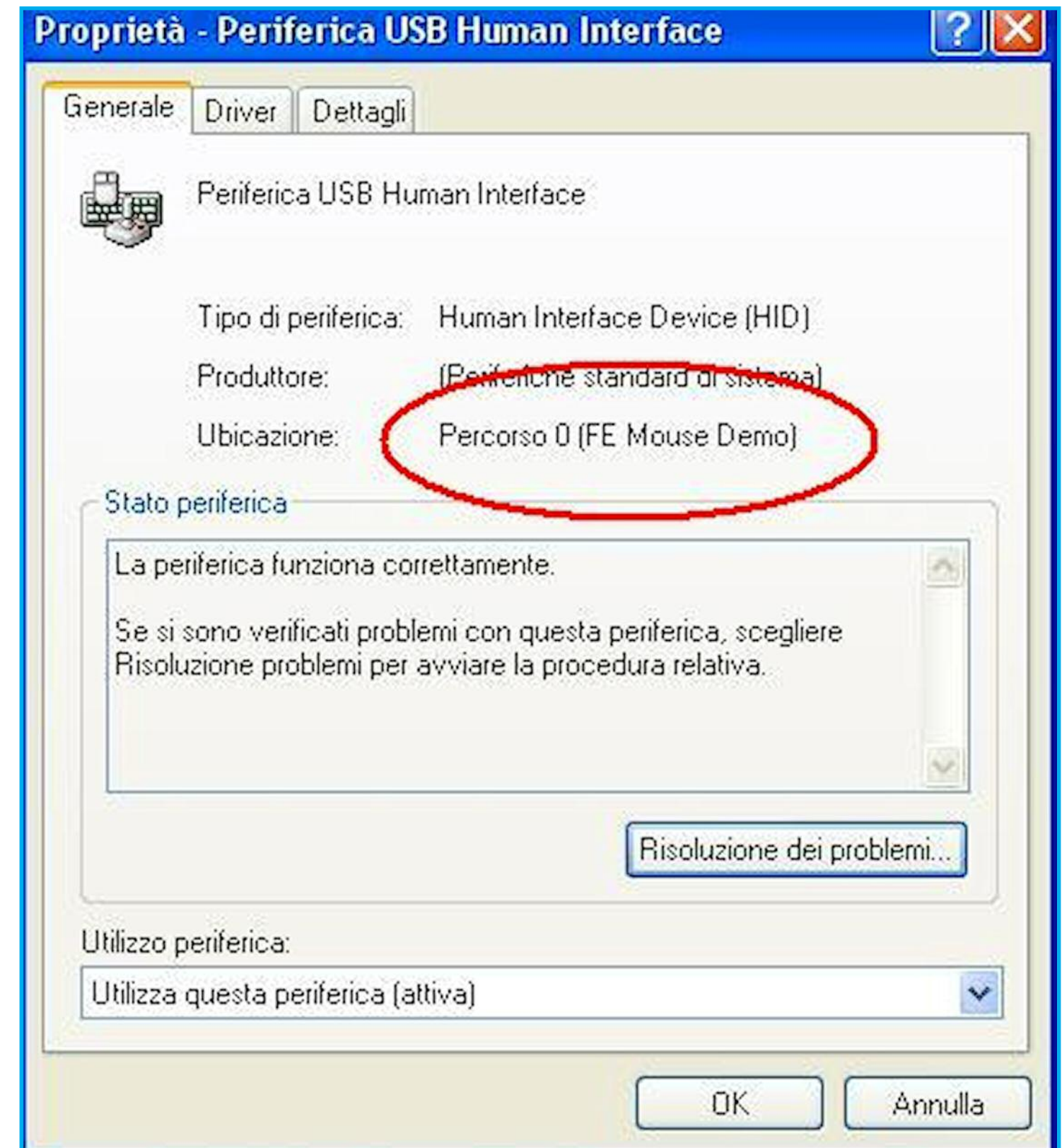



Figura 7 Scheda di descrizione della periferica

pratica, il diver HID lato PC riceverà dal device tre byte, che andrà a decodificare nel modo seguente:

- Il primo byte indica se sono stati premuti dei pulsanti, con la seguente logica:
  - 0, non sono stati prmuti pulsanti
  - 1, è stato premuto il tasto sinistro del mouse
  - 2, è stato premuto il tasto destro del mouse

  Gli altri valori sono riservati ad altre situazioni o a periferiche HID differenti
- Il secondo byte indica se c'è stato movimento sull'asse orizzontale e con che velocità. Il segno indica il senso di movimento (positivo destra, negativo sinistra).
- Il terzo byte indica se c'è stato movimento sull'asse verticale e con che velocità, con la stessa logica utilizzata prima.

Il listato 3 illustra il codice utilizzato per decodificare i pulsanti, per caricare i relativi dati sul buffer e per inviare il tutto sul bus. Con una serie di istruzioni condizionali if..then..else vengono decodificati i pulsanti, poi viene utilizzata la funzione HIDTxHandleBusy, per accertare se è possibile effettuare una nuova trasmissione. Questa funzione restituisce 1 se la precedente trasmissione non è ancora completa (e la periferica è quindi occupata), e 0 nel caso opposto. Se la periferica è pronta i dati contenuti nel buffer temporaneo vengono caricati nel buffer hid_report_in, che costituisce il buffer di trasmissione. A questo punto, tramite l'utilizzo della funzione HIDTxPacket, i dati vengono inviati sul bus. La funzione restituisce inoltre un valore boolenao, 0 nel caso la trasmissione dei dati sia stata completata, 1 nel caso sia ancora in atto.

La funzione HIDTxPacket prende in ingresso tre parametri: l'endpoint hit dal quale vengono inviati i dati, il buffer contenente i dati stessi, e la lunghezza (in byte) dei dati da inviare.

In figura 5 è rappresentato il prototipo utilizzato per l'esecuzione dei test.

CONNESSIONE DEL DISPOSITIVO

Una volta collegato il dispositivo, se tutto è stato fatto correttamente, sentirete il classico "ding" di connessione periferica di windows (o del sistema operativo che utilizzate) e il LED a bordo della scheda inizierà a lampeggiare alternativamente. E' possibile a questo punto visualizzare la periferica tra le risorse hardware di sistema. Per farlo, ad esempio sotto Windows XP, selezionate dal pannello di controllo la voce "Sistema" e poi andate a visualizzare la voce "Gestione periferiche", individuando la sottoclasse "Periferiche compatibili HID", come illustrato in figura 6.

Il Mouse USB che abbiamo creato è stato etichettato con la stringa di descrizione "FE Mouse Demo", come si può vedere nella scheda della periferica riportata in figura 7.

A questo punto la periferica è attiva, e potete utilizzare i pulsanti direzionali per muovere il puntatore e i pulsanti destro e sinistro per effettuare le operazioni tipiche di un mouse. La classe HID può essere utilizzata per implementare anche altre periferiche come tastiere e joystick, ma può essere anche utilizzata come classe generica per la comunicazione da e verso un PC utilizzando il bus USB.

a cura della redazione

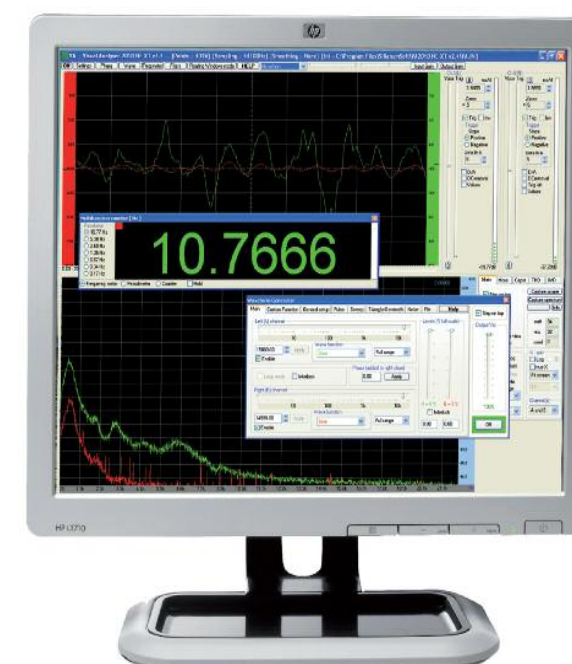
SOFTWARE DA LABORATORIO

Utilizzando un PC ed alcuni software freeware è possibile disporre di una vasta gamma di strumenti da laboratorio utilizzabili nel campo dei segnali di bassa frequenza, senza che sia necessario acquistare ulteriori strumenti "fisici"

Il PC è usato sempre di più per qualunque tipo di misura e in qualunque applicazione di controllo. Esistono hardware e software specifici di diverso tipo pensati appositamente per questo tipo di applicazioni. Del resto, anche gli strumenti di misura standalone prevedono spesso un computer come punto di partenza ed un hardware di I/O come interfaccia oltre ad un software di gestione dedicato appositamente sviluppato. Se non si ha la necessità di eseguire misure su segnali a larga banda o ad alta frequenza è possibile operare efficacemente utilizzando semplicemente il proprio computer ed alcuni software che risultano adeguati e particolarmente utili allo scopo. In questo modo è possibile disporre delle funzioni di un oscilloscopio e di quelle di un generatore di funzione universale, almeno per segnali che rientrano nella banda delle frequenze audio. In questo articolo vedremo software applicativi utili per questi scopi e di tipo freeware e che è possibile quindi utilizzare gratuitamente per scopi privati e didattici (per usi di tipo professionale, cioè a scopo di lucro è necessario sostenere un costo per poter utilizzare il software). In conclusione potete utilizzare strumenti di misura virtuali sostanzialmente gratis – abbiamo bisogno di un computer, ma quello è già disponibile!

DI COSA AVETE BISOGNO?

Effettivamente qualsiasi computer moderno è sufficientemente veloce e potente per effettuare l'acquisizione e la misura o per generare segnali nel campo audio. Ciò di cui abbiamo bisogno è semplicemente un PC su cui far girare applicativi che si basano su sistema operativo Windows. Il Sistema Operativo Windows XP sembra l'opzione migliore da questo punto di vista dal momento che la maggior parte dei programmi di interesse in questo ambito, anche quelli meno recenti, funzionano correttamente senza problemi su Windows XP. Per l'impiego con altre piattaforme meno comuni come Linux e Mac OS la disponibilità di software applicativo di misura e di controllo è purtroppo piuttosto limitata. Ovviamente, le caratteristiche della scheda audio sono essenziali in quest'ambito poiché esse determinano sostanzialmente la capacità di acquisizione e quindi di misura. Fortunatamente, le schede audio negli ultimi anni hanno migliorato mol-

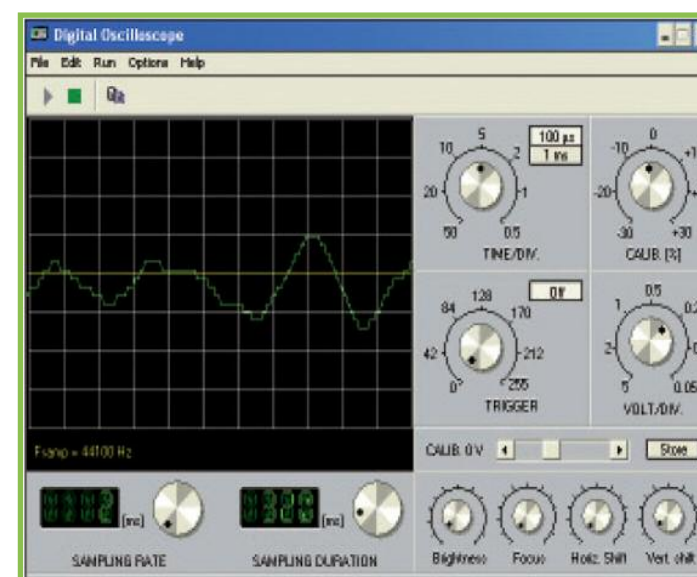
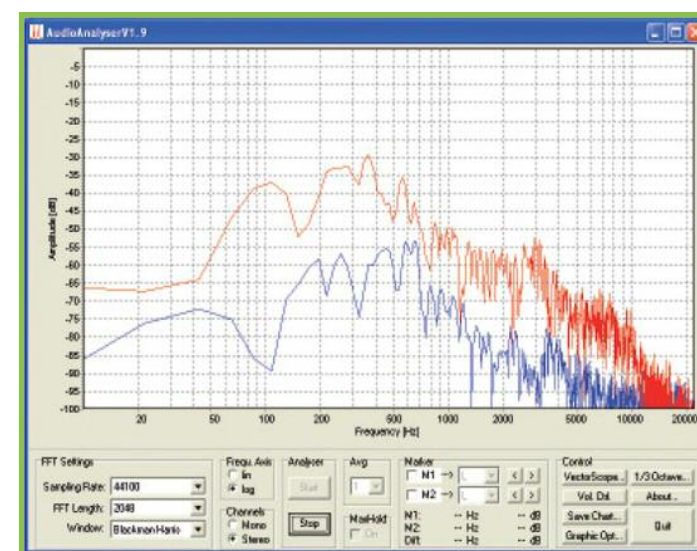
to le loro prestazioni tanto che anche su una scheda madre standard si trova spesso integrato un chip set audio che lavora a 96 kHz. Se avete poi intenzione di comprare una scheda audio dedicata, cioè non integrata nella scheda madre per questi scopi, potreste scegliere un modello con una frequenza massima di campionamento pari a 192 kHz ed una risoluzione pari a 24 bit. Schede di questo tipo per PC sono oggi disponibili a costi che partono da circa un centinaio di sterline.

Gli ingressi e le uscite fisiche di una scheda audio per PC sono di solito costituiti da prese RCA o jack da 3,5 mm. Essi non sono effettivamente adeguati ai nostri scopi cioè nel caso in cui si desidera utilizzarli per acquisire/generare segnali di misura/test. Piuttosto può essere conveniente utilizzare dei cavi adattatori che consentano di passare dal connettore BNC al connettore RCA o al connettore jack da 3,5 mm. Utilizzando tali adattatori è possibile, per esempio, collegare una sonda standard da oscilloscopio per l'acquisizione e la misura dei segnali (nota: usare solo sonde senza attenuazione cioè con rapporto ingresso/uscita 1:1, non utilizzare invece sonde che integrano l'attenuatore 10:1). Lo stesso discorso vale per le uscite: è possibile impiegare un adattatore per utilizzare i comuni connettori BNC, ma è anche possibile realizzare un pratico cavo adattatore con presa a doppia banana.

Quando si utilizza una scheda audio come strumento di misura occorre fare particolare attenzione al valore in tensione del segnale che si intende misurare. Normalmente l'ingresso di linea non è in grado di gestire correttamente segnali più ampi di ± 0,5 V, poiché valori di segnale superiori mandano generalmente in over-drive il convertitore A/D. Di conseguenza, quando è necessario misurare segnali di valore superiore è necessario interporre tra esso è l'ingresso della scheda audio un opportuno attenuatore. A tale proposito occorre anche tenere presente che l'impedenza di ingresso di una scheda audio di questo tipo non è generalmente molto alta essendo tipicamente pari a pochi kΩ. Una scheda audio è quindi chiaramente meno sensibile di uno oscilloscopio reale tanto che essa può potenzialmente influenzare lo stesso circuito sottoposto a misura e quindi il segnale stesso da misurare.

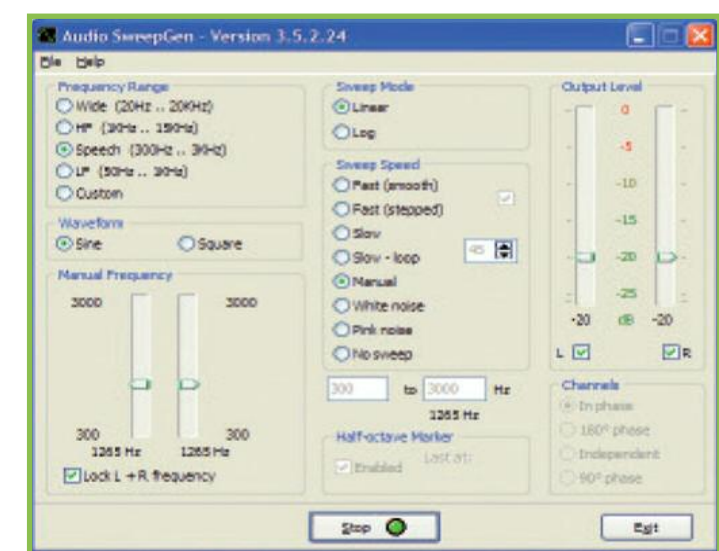
L'ingresso microfonico può essere invece utilizzato per effettuare misure ad un livello di sensibilità abbastanza elevato. E' importante tuttavia osservare che uno dei terminali microfonici presenta una tensione continua necessaria all'alimentazione dei microfoni ad elettrete. E' anche importante osservare che quando si utilizzano le uscite della scheda audio, queste non possono essere caricate eccessivamente; per esempio non è possibile collegare un carico di 50 Ω direttamente all'uscita della scheda audio. In questi casi è pertanto necessario collegare un piccolo amplificatore di potenza alle uscite di linea (utilizzare ad esempio un piccolo amplificatore di un vecchio speaker amplificato per PC e per essere sicuri che sia adeguato è possibile rilevarne la caratteristica in frequenza utilizzando uno dei programmi descritti di seguito).

+Oscilloscopi/Analizzatori Audio

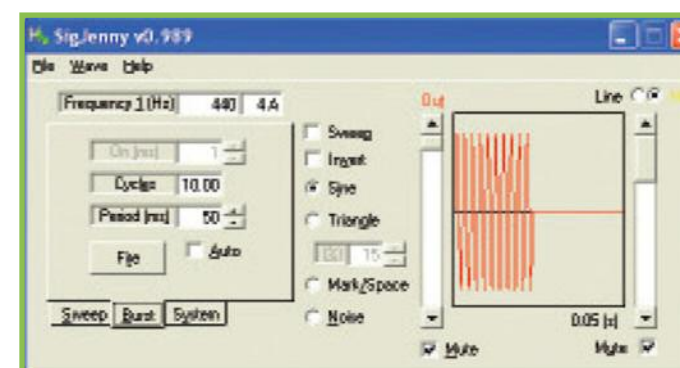
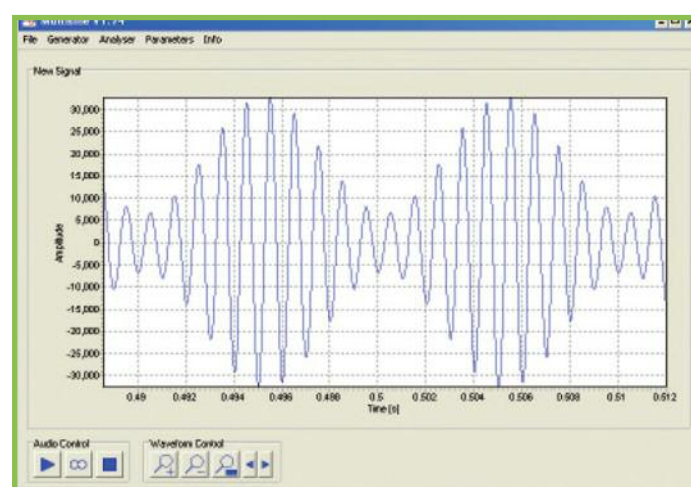
Audio Analyser v1.9 di Sebastian Dunst è un analizzatore di spettro per segnali au-

dio con alcune caratteristiche aggiuntive integrate estremamente utili. Con questo programma freeware è possibile effettuare sul segnale analisi in frequenza in tempo reale. Frequenza di campionamento, lunghezza e tipo di finestra FFT possono essere impostate con estrema facilità. Esso consente di effettuare la media di una serie di misure e di utilizzare due marcatori, che consentono di indicare il livello del segnale a determinate frequenze. Esso integra inoltre numerosi altri utili strumenti: una applicazione che mostra il livello e la differenza di fase tra i segnali dei due canali destro e sinistro rispettivamente ed un analizzatore ad 1/3-ottava. I dati misurati possono anche essere salvati in un file. L'analizzatore FFT reagisce velocemente, in particolare sui computer moderni che sono dotati di notevole potenza di calcolo.

BIP Oscilloscope di Marcel Veldhuijzen è un programma che ha ormai più di dieci anni e, che forse, non offre tutte le caratteristiche e potenzialità offerte da alcuni degli altri programmi descritti in questo articolo. Tuttavia funziona anche con Windows XP, ha un layout chiaro e di semplice utilizzo grazie ai comandi software a forma di manopole rotative. Quando si utilizza questo software sotto Windows XP si ha l'impressione che sia un po' più lento degli altri programmi e che non sia caratterizzato da una riproduzione accurata del segnale. Questo dipende probabilmente dal fatto che si tratta di un software abbastanza datato.

Generatori di funzioni

Audio Sweepgen è un piccolo programma ben strutturato, appositamente sviluppato da David Taylor per produrre sweep audio. E' possibile selezionare la forma d'onda del segnale oggetto dello sweep che può essere sia un'onda quadra che

un'onda sinusoide. Inoltre sono disponibili dei pulsanti per una serie di intervalli di sweep come per esempio quelli del segnale vocale. È possibile selezionare manualmente gli esatti valori di inizio e di fine del range di frequenze di sweep desiderato. La velocità di sweep è completamente regolabile ma è anche possibile scegliere tra alcune impostazioni pre-definite. E' possibile impostare sia sweep lineari che logaritmici ed è anche possibile impostare un marcatore di metà ottava.

BIP Sine Wave Generator è un semplice generatore d'onda sinusoidale, che mette a disposizione due manopole per l'impostazione della frequenza e dell'ampiezza della forma d'onda. E' inoltre disponibile una funzione di sweep. La realizzazione del programma risale alla stessa epoca di Beep scope (ed è anche dello stesso autore) e appare pertanto un po' datato. Esso, tuttavia, funziona ancora su molti computer anche se sul nostro computer abbiamo comunque osservato, di tanto in tanto, strani effetti sul segnale di uscita, che potevano essere eliminati agendo per alcuni attimi con un clic sul pulsante Mute.

Il generatore di funzioni **Multisine V1.74** è un ottimo tool se si vuole generare qualunque tipo di forma d'onda di segnale attraverso una scheda audio. Come suggerisce il nome del tool, questo software può generare forme d'onda di segnale composte, a loro volta, da onde sinusoidali multiple; più precisamente è possibile scegliere la frequenza, l'ampiezza e fase di ciascuna sinusoide componente. Esso consente tuttavia anche di generare tutti gli altri tipi di forme d'onda, come segnali sinusoidali semplici, onde quadre, triangolari o a dente di sega. Inoltre, consente di impostare un "spazzolata" tra due frequenze e di generare un segnale modulato in ampiezza oppure in frequenza. Il software può anche essere impostato per generare un rumore bianco o oppure un rumore rosa. Con l'ausilio dell'analizzatore di spettro integrato è possibile analizzare la composizione in frequenza del segnale generato.

La forma d'onda del segnale di uscita è riportata a video. Un programma davvero versatile!

SigJenny è un programma che funge da generatore di funzioni e che ad una prima impressione potrebbe non sembrare un granché, tuttavia la prima impressione inganna. Ovviamente, è possibile impiegarlo per generare un segnale sinusoidale, triangolare o a dente di sega. Una caratteristica molto interessante è data dal fatto che è possibile trasformare una forma d'onda triangolare lentamente fino a farle assumere l'aspetto di un dente di sega. E' integrata anche la possibilità di eseguire degli sweep in frequenza, sia di tipo lineare che logaritmico, e con evoluzione dello sweep in entrambe le direzioni. Inoltre, esso consente di generare un segnale burst, con la frequenza, il numero di periodi per il quale il segnale perdura e la frequenza di ripetizione impostabili. La forma d'onda generata è mostrata in una piccola finestra dedicata allo scopo. Il programma, infine, offre la possibilità di misurare la caratteristica in frequenza di un altoparlante utilizzando un microfono e di visualizzarla in una apposita finestra. Si tratta di una misura non eccessivamente precisa ma in ogni caso molto utile per una analisi sommaria e rapida.

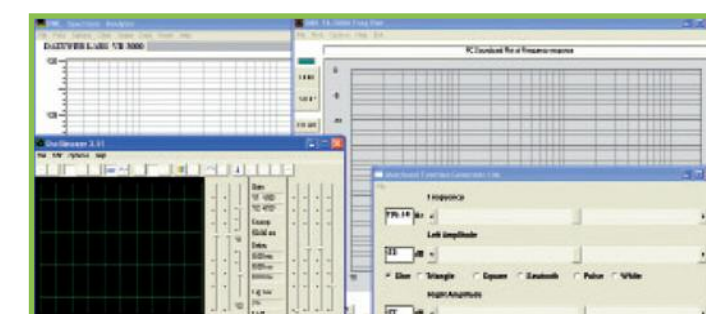
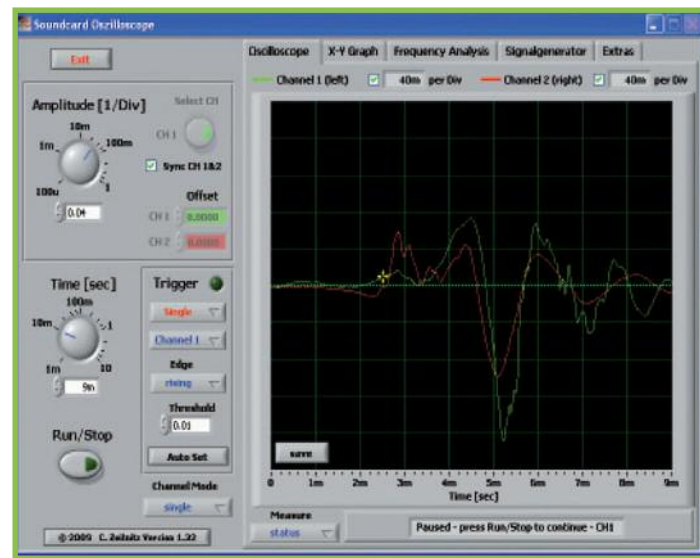
Software Multi-funzioni

Si tratta di programmi applicativi che integrano molteplici funzioni che insieme sono spesso sufficienti a costituire un piccolo e completo laboratorio composto essenzialmente da un oscilloscopio e da un generatore di funzioni.

Audio Test Bench è una raccolta di singoli programmi utilizzabili gratuitamente e scaricabili da HigherFi.com, un rivenditore on-line di apparecchiature audio di alto livello. La raccolta comprende, tra gli altri, un pratico oscilloscopio che originariamente era disponibile come applicativo freeware sviluppato diversi anni fa da uno studente russo e che ora, noto nella nuova versione chiamata Zelscope, può essere acquistato ad un costo piuttosto contenuto.

La collezione comprende anche un analizzatore di spettro che deriva da Dazyweb Labs, un semplice generatore di toni in cui è possibile impostare la frequenza, l'ampiezza e diverse forme d'onda di segnale e infine un analizzatore di spettro che è possibile utilizzare per ricavare una caratteristica in frequenza e che deriva ancora da DazyWeb.

Si tratta di software abbastanza datati, ma che costituiscono comunque una bel-

la collezione da tenere a portata di mano quando si vuole misurare qualcosa.

Soundcard Scope è una bella applicazione di misura realizzata interamente in Lab-View. Il programma appare proprio come un oscilloscopio, con pulsanti che è possibile comandare attraverso il mouse per impostare la scala del segnale in ingresso e la base de tempi. Il programma reagisce velocemente alle variazioni del segnale di ingresso, tanto da dare la reale sensazione di lavorare con un vero oscilloscopio.

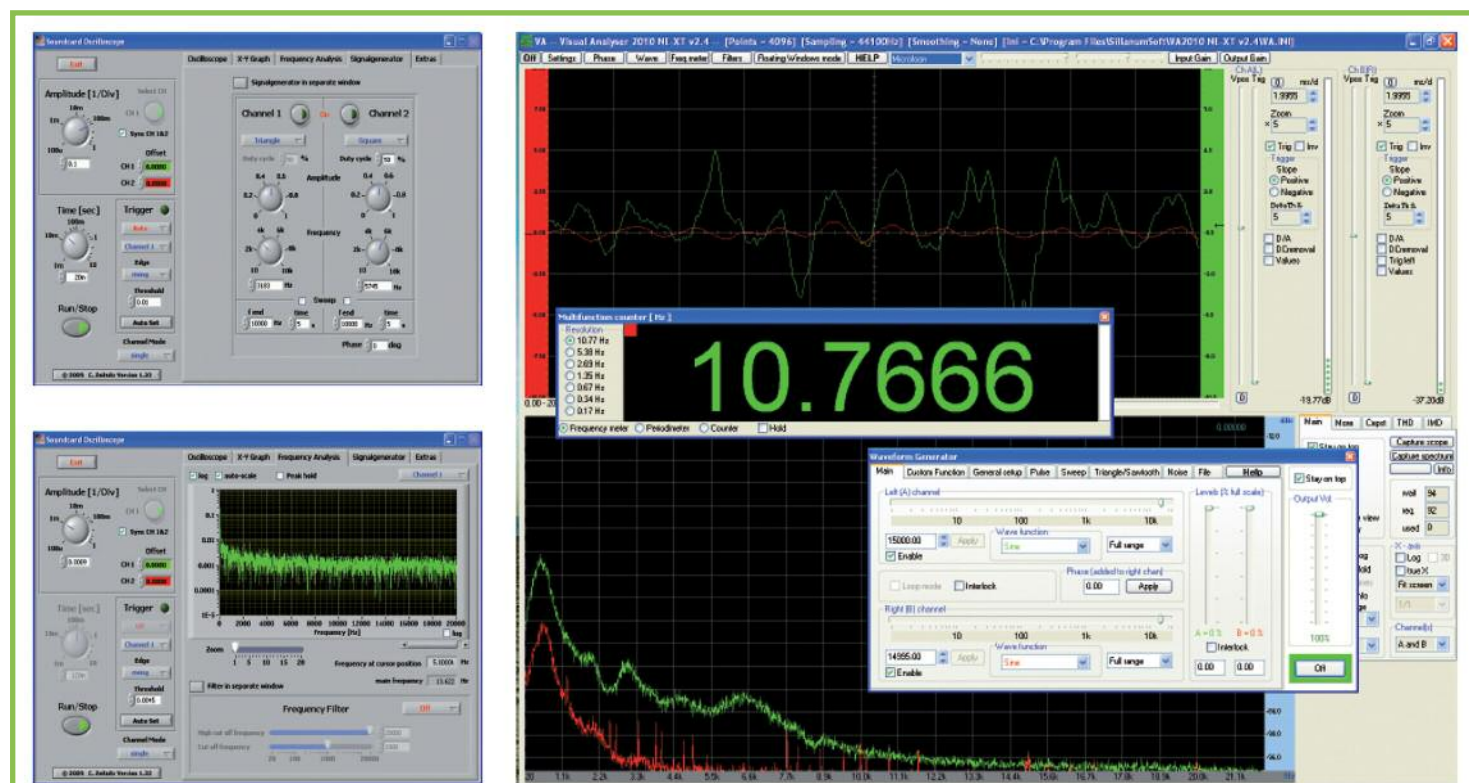
Sono disponibili tutte le funzioni tipiche di un oscilloscopio comprese diverse opzioni di trigger. Oltre alla visualizzazione classica del segnale è possibile utilizzare appositi tab e quindi pagine per avviare una modalità di rappresentazione di tipo XY per la rappresentazione delle figure di Lissajous. Inoltre vi è la possibilità di visualizzare una finestra in cui viene mostrata l'analisi in frequenza (FFT) cioè l'analisi di Fourier del segnale d'ingresso acquisito. Il tab successivo e la relativa pagina rendono disponibili un generatore di funzioni a due canali, che può essere impostato attraverso tasti rotativi.

Il segnale generato può appartenere a diverse forme d'onda come seno, triangolare, quadrata, a dente di sega oppure può essere un rumore bianco. E' anche possibile impostare sul segnale generato uno sweep tra due frequenze. Accedendo all'ultima pagina integrata nell'interfaccia software è possibile agire sulle impostazioni della scheda audio (per esempio sulla frequenza di campionamento) ed impostare la registrazione del segnale sottoposto a misura e la sua memorizzazione all'interno di un file .wav. Davvero utile!

Visual Analyzer è un programma realizzato in Italia caratterizzato da un numero davvero ampio di funzioni. Per default l'interfaccia mostra sullo schermo due ampie finestre di lavoro di cui una funge da oscilloscopio a due canali e l'altra mostra simultaneamente l'analisi FFT dei segnali misurati.

Sia la funzione da oscilloscopio che quella di analizzatore FFT reagiscono prontamente e rendono pertanto l'interazione ed il lavoro molto piacevole. Esso risulta dotato di numerose impostazioni per cui, a volte, individuare l'opzione più adeguata può sembrare non immediato. Inoltre è possibile visualizzare la caratteristica di fase in una finestra separata e anche aprire una apposita finestra che integra un frequenzimetro.

Naturalmente, è anche integrato un generatore di segnale completo in quanto in grado di generare diverse forme d'onda e di gestire anche opzioni di sweep in frequenza. Altre caratteristiche tipiche di questo tool sono la possibilità di calcolare automaticamente la THD e la possibilità di effettuare misure LCR con l'aiuto di alcuni componenti hardware aggiuntivi (gli schemi relativi sono disponibili sul sito dell'autore).

Purtroppo il design può sembrare un po' disordinato e il programma non sempre segue le convenzioni standard di Windows, ma se si accetta questo si tratta di un programma che offre davvero una serie notevole di caratteristiche e di funzionalità.

Per approfondire

[1] Audio Analyser: http://softsolutions.sedutec.de/audioanalyser.php

[2] BIP Oscilloscope: www.electronics-lab.com/downloads/pc/002/index.html

[3] Audio Sweepgen: www.satsignal.eu/software/audio.html#SweepGen

[4] BIP Sine Wave Generator: www.electronics-lab.com/downloads/pc/005/index.html

[5] Multisine: http://softsolutions.sedutec.de/multisine.php

[6] SigJenny V0.989: www.natch.co.uk/downloads/SigJenny/SigJenny.html

[7] Audio Test Bench: www.higherfi .com/software.htm

[8] Soundcard Scope: www.zeitnitz.de/Christian/scope_en

[9] Visual Analyzer: www.sillanumsoft.org/prod01.htm

IEshop

RACCOLTA ANNATE FARE ELETTRONICA 2003/2012

Tutte le riviste uscite dal 2003 al 2012 in formato pdf raccolte in comodissimo file zip da scaricare. In più avrete tutti I downloads relativi ai singoli articoli, i numeri speciali ed una interfaccia grafica per navigare attraverso I contenuti

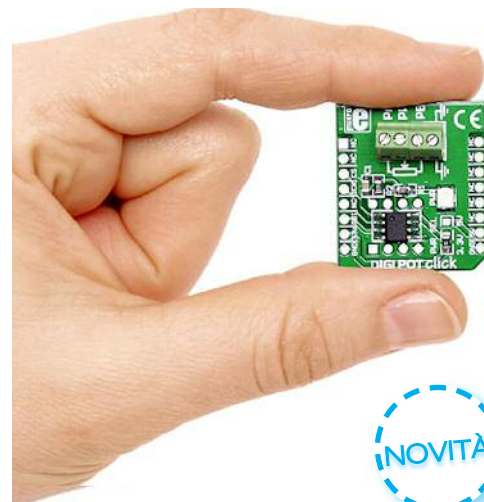
Prezzo: € 20.16

GPS CLICK

GPS Click - L10 è una scheda accessoria che utilizza il nuovo socket mikroBUS. Si tratta di una soluzione compatta per l'aggiunta di funzionalità GPS sul vostro dispositivo. È dotato di Quectel ® L10, motore di posizionamento MTK ad alte prestazioni. La scheda può essere interfacciata con un microcontrollore tramite connessione UART o I2C, oppure i dati possono essere acquisiti utilizzando una applicazione applicazione per PC attraverso la connessione USB. Una delle caratteristiche della scheda è che il connettore è compatibile con le antenne attive e passive. Funziona solo con alimentazione a 3.3V.

Prezzo: € 47.19

POTENZIOMETRO DIGITALE

DigiPot Click è una scheda accessoria che utilizza il nuovo socket mikroBUS. È dotata di un potenziometro digitale singolo MCP4161 con risoluzione a 8 bit (256 step) e l'interfaccia seriale standard SPI. Il valore della resistenza del potenziometro digitale arriva fino a 10 kohm. È dotato di notevoli caratteristiche di AC/DC e un basso consumo di potenza. Può essere utilizzato in apparecchiature audio, controllo servomotore, controllo di carica batterie, controllo regolazione contrasto LCD, filtri programmabili ecc

Prezzo: € 16.36

4-20 MA T CLICK

4-20mA T Click è una scheda accessoria che utilizza il nuovo socket mikroBUS. Si tratta di una soluzione compatta e semplice per l'aggiunta della comunicazione standard 4/20mA nel vostro progetto. È dotato dell'isolatore digitale Quad-chanel ADuM1411, del DAC a 12 bit MCP4921, nonché del trasmettitore del loop di corrente 4/20mA XTR116. La scheda 4-20mA T Click comunica con il microcontrollore target attraverso tre linee SPI su mikroBUS (SDI, SCK CS). La scheda è progettata per utilizzare alimentazione a 3,3 V e 5V. Un diodo LED indica la presenza di alimentazione.

Prezzo: € 25.17

MICRO-OLED 96X64 + INTELLIGENT CONTROLLER, GOLDELOX PROCESSOR

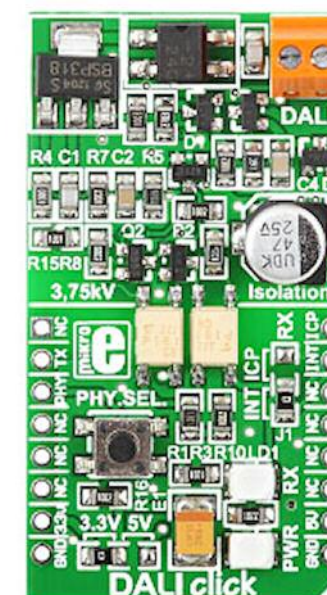
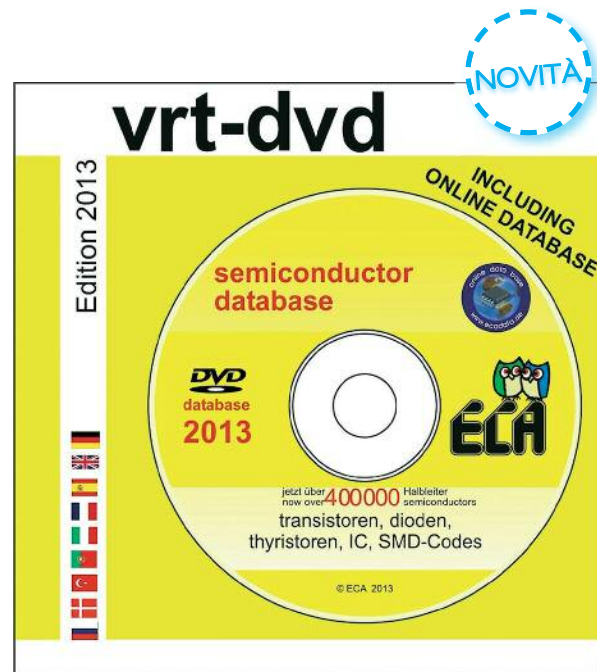
OFFERTA

Il microOLED μOLED-96 è un display grafico a colori completo di scheda di controllo che può essere gestito direttamente da un microcontrollore (PIC, AVR, ARM, Stamp) via porta seriale. Possibilità di scegliere 3 tipi di dimensioni per il font (5x7, 8x8 and 8x12) e con appositi comandi disegnare cerchi, linee, rettangoli, scegliere il colore dello sfondo e caricare immagini bitmap. A bordo adattore memory card per micro-SD (μSD) per immagazzinare icone, immagini, animazioni ecc..

Prezzo: ~~€ 47.19~~ € 35.09

DALI CLICK

DALI Click è una scheda accessoria che utilizza il nuovo socket mikroBUS che fornisce il Digital Addressable Lighting Interface (DALI) per il vostro progetto. La scheda dispone di due optoaccoppiatori, pulsantie e terminali a vite. La DALI Click comunica con la scheda di destinazione attraverso quattro linee mikroBUS (RST, CS, PWM e INT). I dati tra MCU e i dispositivi vengono trasferiti attraverso il bus differenziale a due fili tramite il protocollo seriale asincrono half-duplex. E ' possibile indirizzare fino a 64 dispositivi DALI stand alone e più di 64 dispositivi DALI come sottosistemi (gateway). La scheda è progettata per utilizzare alimentazione a 3.3V o 5V. Un diodo LED indica la presenza di alimentazione.

Prezzo: € 24.20

DATABASE COMPLETO PER TUTTI I TIPI DI SEMICONDUTTORI SU DVD-ROM

ECA VRT-DVD 2013 è il database più completo per tutti i tipi di semiconduttori su DVD-ROM con software per Windows® di facile utilizzo.

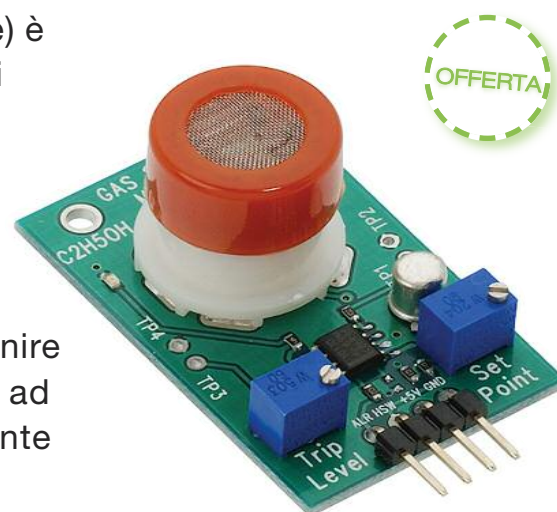
Oltre la ricerca e selezione per tipo, dispositivo, produttore, outline case, codice SMD e testo descrittivo per tutti i componenti, permette la ricerca parametrica di transistori, diodi, tristori, triac, UJT, DIAC, tetrodi e FET.
La nuova versione aggiornata 2013 contiene i dati di circuiti integrati, transistori, diodi, FET, tristori, UJT, amplificatori operazionali, componenti di gestione elettrica, convertitori DC-DC, switch, regolatori di voltaggio etc, tutti con importanti dettagli nonché l'accesso ai datasheet originali in formato PDF (se disponibili). Inoltre, è possibile salvare note pubbliche e private sui server.
Contenuto:

- oltre 98.000 diodi
- oltre 75.000 FET
- oltre 138.000 transistori
- oltre 18.000 tristori
- oltre 135.000 circuiti integrati

Questo nuovo DVD include il database aggiuntivi dei CD-ROM tdv-disk e ddv-disk. Inoltre, tutti gli utenti registrati al software avranno accesso aggiuntivo per un anno al database online vrt (circuiti integrati), tdv (transistori, FET) e ddv (diodi, tristori).

 Prezzo: € 62.92

MODULO SENSORE GAS

Il Modulo Sensore Gas C2H5OH (Alcohol/Benzine) è disegnato per permettere a un microcontrollore di determinare il momento in cui il livello di gas C2H5OH precedentemente settato è stato raggiunto o superato. Il collegamento al modulo sensore è operato grazie a un SIP header 4-pin e richiede 2 pins I/O dal microcontrollore host. Il modulo sensore è principalmente indirizzato a fornire un mezzo di comparazione delle fonti di alcohol e ad essere in grado di settare un allarme qualora la fonte eccede.

 Prezzo: ~~€ 34.79~~ € 22.99

MICROLCD 3.2" PICASO DISPLAY MODULE CON TOUCH RESISTIVO

Il LCD-32PTU è un modulo Display Intelligente compatto ed economico dotato di tantissime funzioni, pronto a diventare interfaccia grafica per alcune applicazione target e in grado di essere una interfaccia di controllo per altre applicazioni. Il cuore di questo modulo è il processore PICASO, che è guidato da un virtual core engine altamente ottimizzato: EVE (Extensible Virtual Engine). Una vasta gamma di periferiche hardware e software sono stati integrati in questo modulo, in maniera tale da dare all'utente la libertà di adattare il modulo per ogni qualsiasi applicazione. L' LCD-32PTU è una elegante combinazione di uno schermo LCD 2,2 "(240x320), amplificatore audio e altoparlante, connettore micro-SD card, supporto a batteria ai polimeri di litio (LiPo), insieme ad un gruppo di pin I/O general purpose (GPIO), compresi I2C e COMMS seriale. Il modulo display LCD-32PTU funge da perfetta soluzione per progetti che richiedono una brillantezza di colore, animazione o immagini su qualsiasi applicazione. Questo modulo Display Intelligente guidato da PICASO è un perfetto esempio dove l'arte incontra la tecnologia.

 Prezzo: € 76.23

MODULO 3-ASSI PER RILEVAMENTO DI CAMPI MAGNETICI

Il Compass Module 3-Axis HMC5883L è stato progettato per il rilevamento di bassi campi magnetici con interfaccia digitale. Questo sensore, molto compatto, si adatta in progetti di piccole dimensioni come UAV e sistemi di navigazione di robot. Il sensore converte ogni campo magnetico a una tensione di uscita differenziale su 3 assi.
Questo cambiamento di tensione è il valore grezzo di uscita digitale, che può quindi essere utilizzato per calcolare le direzioni o rilevare campi magnetici provenienti da direzioni diverse. Il codice di esempio in PBASIC, Spin, e C è disponibili nel download

 Prezzo: € 30.25

DUAL DC MOTOR SHIELD PER ARDUINO

La Dual DC Motor Shield è una shield progettata per permettere ad una scheda Arduino di pilotare due motori DC o un motore passo-passo bipolare, controllando sia la velocità sia la direzione di ciascun canale in modo indipendente.
Alcune applicazioni possono richiedere di controllare la velocità del motore o la posizione angolare di un servo tramite potenziometro; per questi motivi in questa shield sono state aggiunte due porte digitali di I/O e due porte di ingresso analogiche. Le porte analogiche possono essere usate per leggere potenziometri o altri sensori analogici, mentre le porte di I/O digitali sono utilizzabili per pilotare servomotori o interfacciarsi ad altri dispositivi digitali.
Questa shield è basata sul noto circuito integrato L298 prodotto dalla STMicroelectronics, un doppio driver a ponte-H ad alta tensione ed alta corrente. Per prevenire il danneggiamento della scheda, le linee di uscita del driver hanno diodi di protezione contro le back EMF. La massima tensione supportata è 46V.
Per garantire condizioni operative di sicurezza è possibile misurare la corrente assorbita da ciascun motore; questo permette di evitare condizioni di stallo. E' inoltre presente un sensore digitale di temperatura integrato con interfaccia I2C utile per monitorare la temperatura della shield con lo scopo di prevenire il pericolo di surriscaldamenti.

Prezzo: € 24.44

ANNATA RIVISTA FIRMWARE 2012

Un file zip che contiene tutte le riviste del 2012 in formato pdf ad alta risoluzione, tutti i downloads realtivi ai singoli numeri. E' incluso anche Acrobat Reader 6 per sfogliare le riviste.

Prezzo: € 9.99

- Sei già cliente? Risparmierai il 10% sul tuo nuovo ordine!

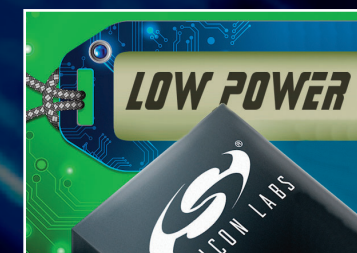
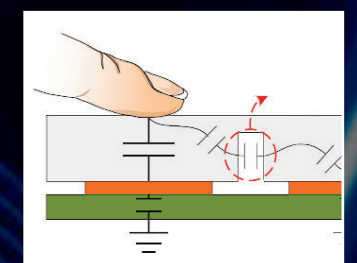
- Il tuo ordine supera i 200 EUR? Le spese di trasporto sono Gratis!

- Vuoi ricevere un buono sconto di 5 EUR? Recensisci i prodotti acquistati!

- Vuoi ricevere un coupon del 10% di sconto? Rispondi ai quesiti di Elettroquiz!

- Vuoi ricevere particolari offerte o promozioni? Diventa membro Inware Edizioni.

MOTORI BRUSHLESS

SOLUZIONI E-FIELD PER IL CONTROLLO DELL'ILLUMINAZIONE

SCEGLIERE UN MICROCONTROLLORE LOW-POWER

SENSORE CAPACITIVO TOUCH

Protocolli per il lighting

DALI e DMX

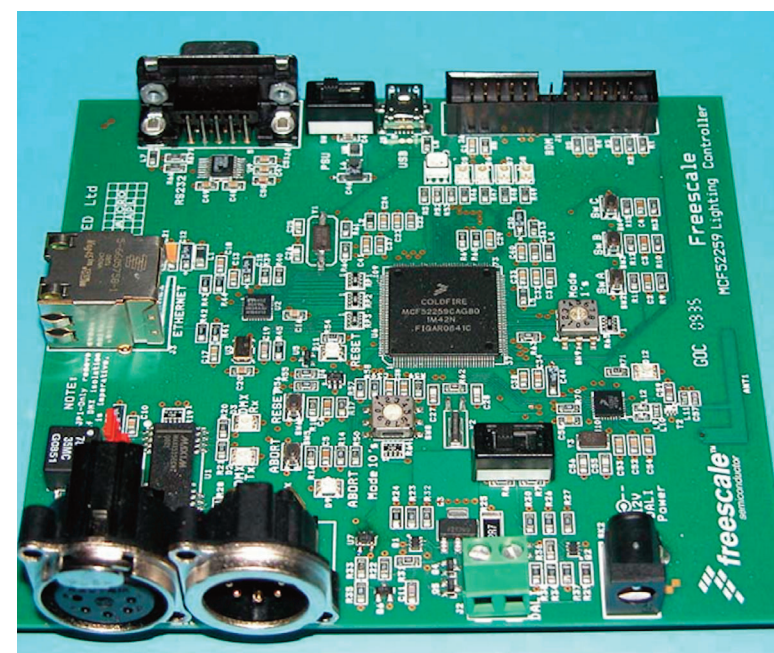
Lighting & Lighting

EDITORIALE

Uno dei primi schemi elettrici che ho visto da giovane studente è stato lo schema dell'accensione di una lampadina da un punto, per poi analizzare l'accensione da due punti e infine l'accensione della stessa lampada da 3 o più punti. Tutto molto semplice: interruttori, deviatori, invertitori... e in caso di non funzionamento era piuttosto semplice risalire all'origine del problema. Oggi la complessità degli impianti elettrici negli edifici ha raggiunto livelli tali da rendere quasi impossibile il cablaggio tradizionale. Addio quindi a interruttori, deviatori e invertitori e benvenute nuove tecnologie. Una di queste è "DALI", acronimo di "Digital Addressable Lighting Interface", un protocollo di comunicazione standardizzato dalla normativa CEI EN62386. E proprio il "DALI" è l'argomento del Focus On di questo numero. L'articolo non ha l'ambizione di essere un trattato completo sull'argomento, ma senz'altro un buon punto di partenza per chi intende approfondire l'argomento. Sempre in tema Lighting, in questo numero troverete le soluzioni Freescale basate su NFC, anche in questo caso una grande innovazione che contribuirà a trasformare gli interruttori meccanici in un lontano ricordo.

E-Field Lighting Controller

Le proposte innovative di Freescale per il dimming

BOARD HIGH-BRIGHTNESS

"Networked LED lighting" mediante i protocolli DALI e DMX512A

+Skills

MICRONTROLLORI LOW POWER: COME SCEGLIERE

SOLUZIONI USB POWER PER BATTERIE AI POLIMERI DI LITIO (LIPO)

E-FIELD Lighting Controller

Le proposte innovative di Freescale per il dimming

di VINCENZO GERMANO

Citando la presentazione dell'E-Field Lighting Controller sul sito della Freescale, si legge: "In futuro, gli interruttori meccanici per il controllo dell'illuminazione saranno qualcosa del passato", infatti ad oggi la diffusione di sensori touch capacitivi di prossimità sta sempre più prendendo piede. Indipendentemente se essi siano '*slider* ' lineari o circolari, questi offrono comunque molti vantaggi nelle applicazioni per il controllo dell'illuminazione casalinga, perché sono convenienti da usare non avendo parti mobili che si usurano, sono molto efficienti da un punto di vista energetico e sono facili da integrare in un sistema di rete domestica. Freescale Semiconductor dimostra questi e altri vantaggi implementando una soluzione che combina uno slider circolare (associato a un sensore di prossimità) con un modulo ZigBee® SMAC (Simple Media Access Controller) per il controllo wireless di varie prese intelligenti e la gestione del dimming di lampadine a incandescenza standard, fornendo in dettaglio tutto il materiale del caso, incluso hardware e software come schematici, bill of material, file gerber per le PCB e altro ancora. Ma analizziamo il sistema nel dettaglio.

Figura 1: System Overview

SYSTEM OVERVIEW

L'illuminazione domestica mediante lampade o lampadine ad oggi è uno standard ma Freescale offre una soluzione innovativa e di comfort aumentando/migliorando le caratteristiche di tale applicazione. In Figura 1 è riportato il sistema completo proposto da Freescale, in cui si distinguono principalmente due parti:
"Switch intelligenti" (SI) che possono sostituire i classici interruttori a muro (ne è visibile uno al centro della Figura 1), generando la funzione di dimming sulla fase dell'AC (Alternating Current – corrente alternata). L'interfaccia di controllo è un pannello senza contatto (mediante l'uso di campo elettrico – E-field contactless) con la possibilità di scegliere tra varie forme di elettrodi. Questi SI possono tranquillamente gestire quattro prese intelligenti mediante un trensceiver radio a bassa potenza nella banda IMS dei 2.4 GHz utilizzando il SMAC basato sullo Zigbee™. Prese intelligenti (PI) che sostituirebbero le classiche prese da muro e gestendo direttamente l'AC, effettuano un dimming della luce delle lampade standard o di dispositivi a esse direttamente collegati. Freescale fornisce una flessibilità aggiuntiva rendendo disponibili queste prese in due implementazioni differenti: presa a muro (ne sono visibili tre sul lato sinistro della Figura 1) e adattatore

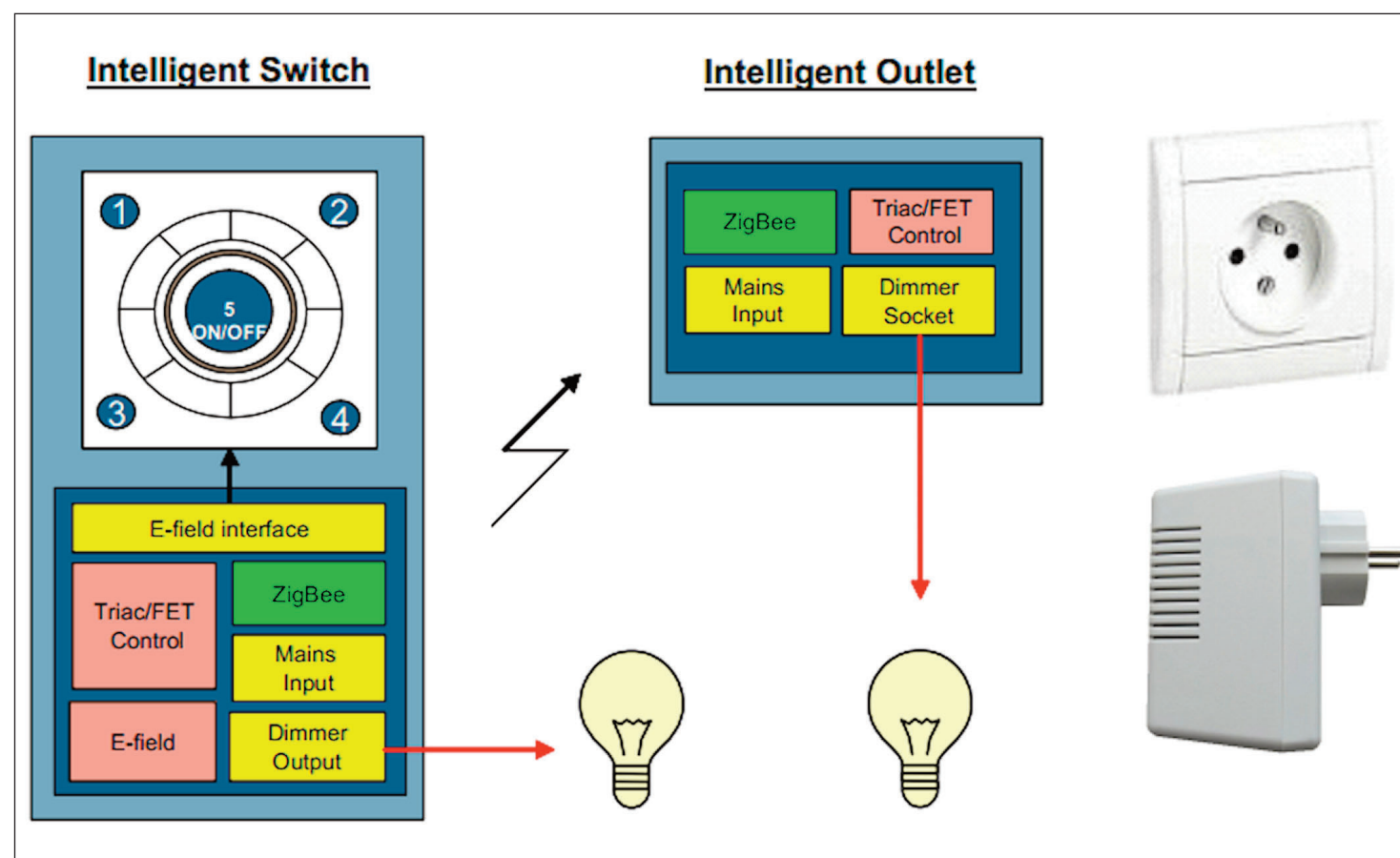



Figura 2: Blocchi funzionali del sistema

da collegare a muro (ne sono visibili tre sul lato destro della Figura 1). Per meglio comprendere l'architettura di sistema si faccia riferimento alla Figura 2, in cui sono presenti gli schemi a blocchi di entrambe le parti sopraelencate. L'applicazione utilizza un concetto stand-alone, nella scheda di controllo principale, il modulo SI, è presente un microcontrollore per la gestione del sistema e un sensore E-Field come interfaccia verso l'esterno. Perciò le prese intelligenti sono controllate dallo switch intelligente che fornisce l'interfaccia di controllo (Control Panel). Le schede sono alimentate direttamente dalla rete AC e le funzionalità dei moduli possono essere modificate tramite il software e la configurazione hardware sul PCB. Di tutto questo sistema si può affermare che uno dei vantaggi principali è la comoda interfaccia di controllo realizzata in tecnologia E-Field. Gli elettrodi sono situati sotto il pannello a sfioramento e sono visibili come blocco funzionale in Figura 2 (nella parte alta degli switch intelligenti, contrassegnati con i numeri da 1 a 5), in cui è anche riportato il blocco funzionale d'interfaccia interna verso il microcontrollore

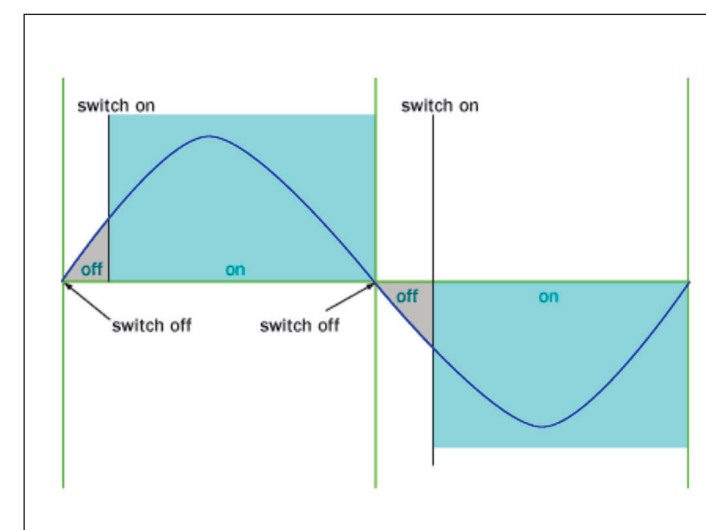



Figura 3: Funzionamento del processo switching mediante Triac

(E-field interface). Le forme (a scorrimento o circolari) e le posizioni di questi elettrodi possono essere completamente personalizzate in base alle esigenze. L'altro grande vantaggio è la comunicazione senza fili RF (Radio Frequenza). Il modulo Zigbee è comune sia agli SI che alle PI (in Figura 2 blocco verde) e fornisce molte possibilità per la realizzazione della rete wireless. Un dispositivo può comunicare a un livello peer-to-peer con altri dispositivi, come "master" quando si controllano più dispositivi contemporaneamente oppure mediante un livello gerarchico "slave". Nella sottorete, i dispositivi sorgente o destinatari possono essere definiti nell'intestazione del pacchetto di protocollo e tutto dipende dalla gestione del protocollo individuale, facilmente configurabile dall'utilizzatore finale. Il

ricetrasmettitore SMAC basato su ZigBee opera nella banda di frequenza da 2,4 GHz ISM, che garantisce la non interferenza con le apparecchiature di casa standard. Inoltre, la scheda può essere utilizzata come una piattaforma hardware per lo sviluppo software. A questo scopo, la scheda è equipaggiata con l'interfaccia di programmazione BDM (Debug Background Mode). Questo strumento permette alla memoria MCU (microcontrollore) di essere riprogrammata facilmente in-circuit. In Figura 2 per entrambe i moduli si distinguono: il blocco del "Mains Input" che corrisponde alla gestione in ingresso dell'AC; il blocco del "Dimmer Socket/Output" che riguarda l'interfaccia di uscita per le gestione lampade o device esterni e infine "TRIC/Fet Control" relativo al blocco per la generazione del dimming. I maggiori vantaggi di questa soluzione proposta da Freescale sono:

Ridotto consumo di energia e cost saving;
Eliminata la necessità di acquisto di lampade costose;
Ridurre il numero di volte in cui è necessario cambiare le lampadine;
Facilità di adattare il software alle varie lampade, dipendentemente dalla loro potenza;
Il dimmer supporta tensioni di 230Vac/50Hz e 115Vac/50Hz;

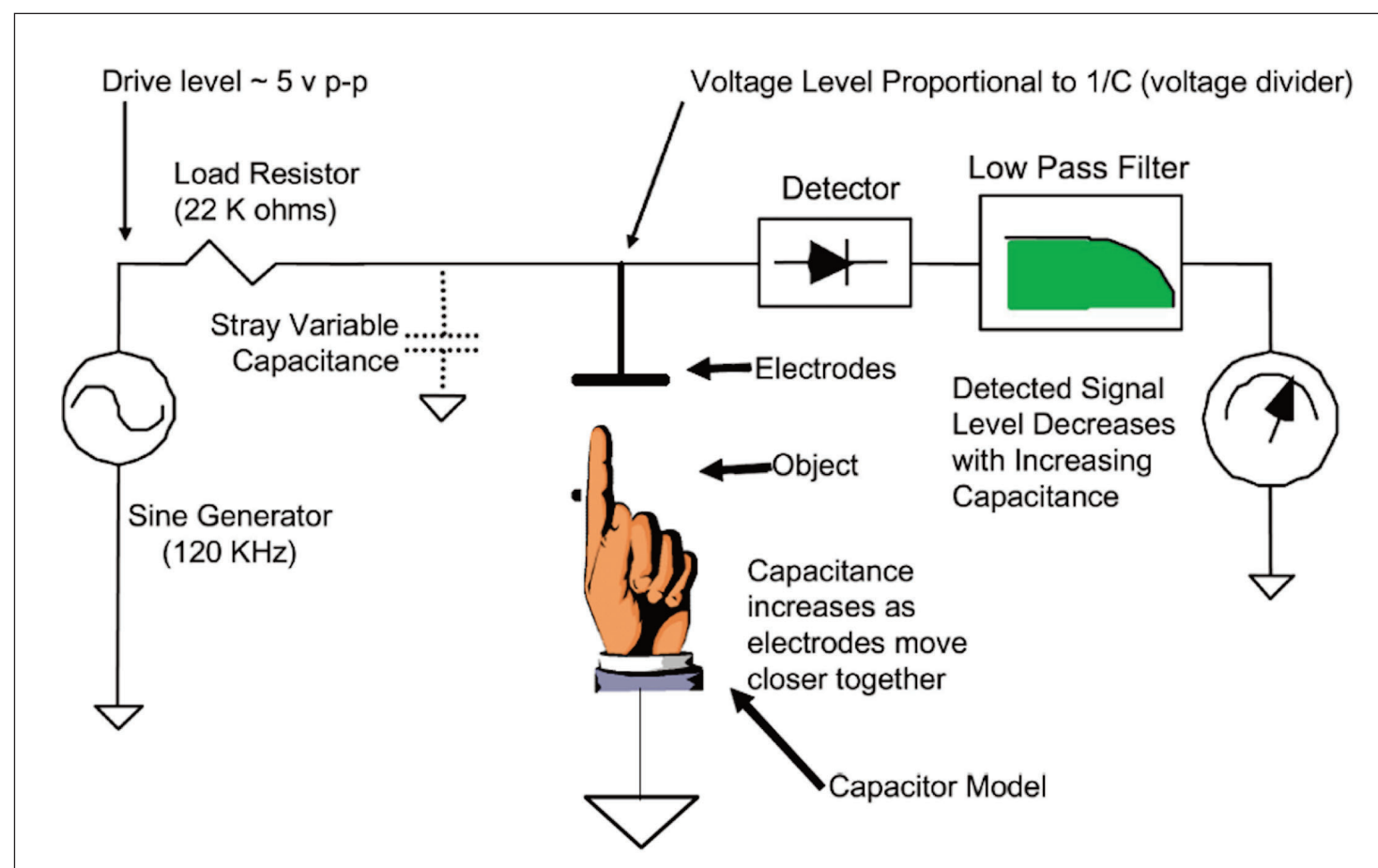



Figura 4: Principio di funzionamento

Facilità di ri-programmare il comportamento del sistema mediante software semplifica l'hardware;
L'interfaccia di controllo E-field senza contatto offre varie possibilità di materiali per il design della cover frontale;
Interfaccia di controllo completamente variabile (numero, dimensioni, forme degli elettrodi E-Field);
Apertura verso l'innovazione.

Per una maggiore comprensione, prima di addentrarci nella trattazione delle varie parti del sistema analizziamo brevemente la teoria di funzionamento del dimming e il funzionamento dei control panel.

TEORIA DI FUNZIONAMENTO: IL DIMMING

In generale le varie stanze di una casa sono usate per diverse funzioni che possono richiedere differenti quantità di luce e mediante il dimming si riescono a regolare i livelli di luce di una lampadina, da quasi buio a completamente illuminata, semplicemente ruotando una manopola o attraverso lo scorrimento di un controller. Esistono varie tipologie di circuiti per il dimming ma in generale le vecchie configurazioni deviavano l'energia della lampada in un resistore variabile mentre quelli moderni spengono e accendono rapidamente il circuito d'illuminazione per ridurre la quantità totale di energia che fluisce attraverso la lampada, il tutto realizzato e concepito direttamente per la corrente alternata (AC). In Figura 3 è riportato il funzionamento concettuale di tali sistemi; come si può notare i nuovi dispositivi automaticamente spengono la luce della lampadina ogni volta che la direzione della corrente si inverte e la tensione attraversa lo zero (quindi due volte per ciclo e cento volte al secondo), riaccendendole quando il valore di tensione (turn-on) risale oltre una certa soglia prestabilita. Tale valore dipende dalla posizione dell'interfaccia di controllo (es. la manopola), infatti nel caso in cui è rivolta maggiormente verso l'impostazione di alta luminosità, lo "*switch on*" visibile nella figura precedente è molto vicino all'attraversamento dello zero da parte della tensione e quindi il circuito rimarrà acceso per la maggior parte del tempo fornendo alla lampadina una maggiore energia; mentre nel caso in cui l'interfaccia di controllo è rivolta maggiormente verso un'impostazione di bassa luminosità, lo "*switch on*" sarà più distante dallo zero e la lampadina rimarrà spenta per un tempo maggiore rispetto al precedente. L'implementazione di questo funzionamento nei circuiti odierni viene realizzata mediante l'utilizzo di un Triac, per con-

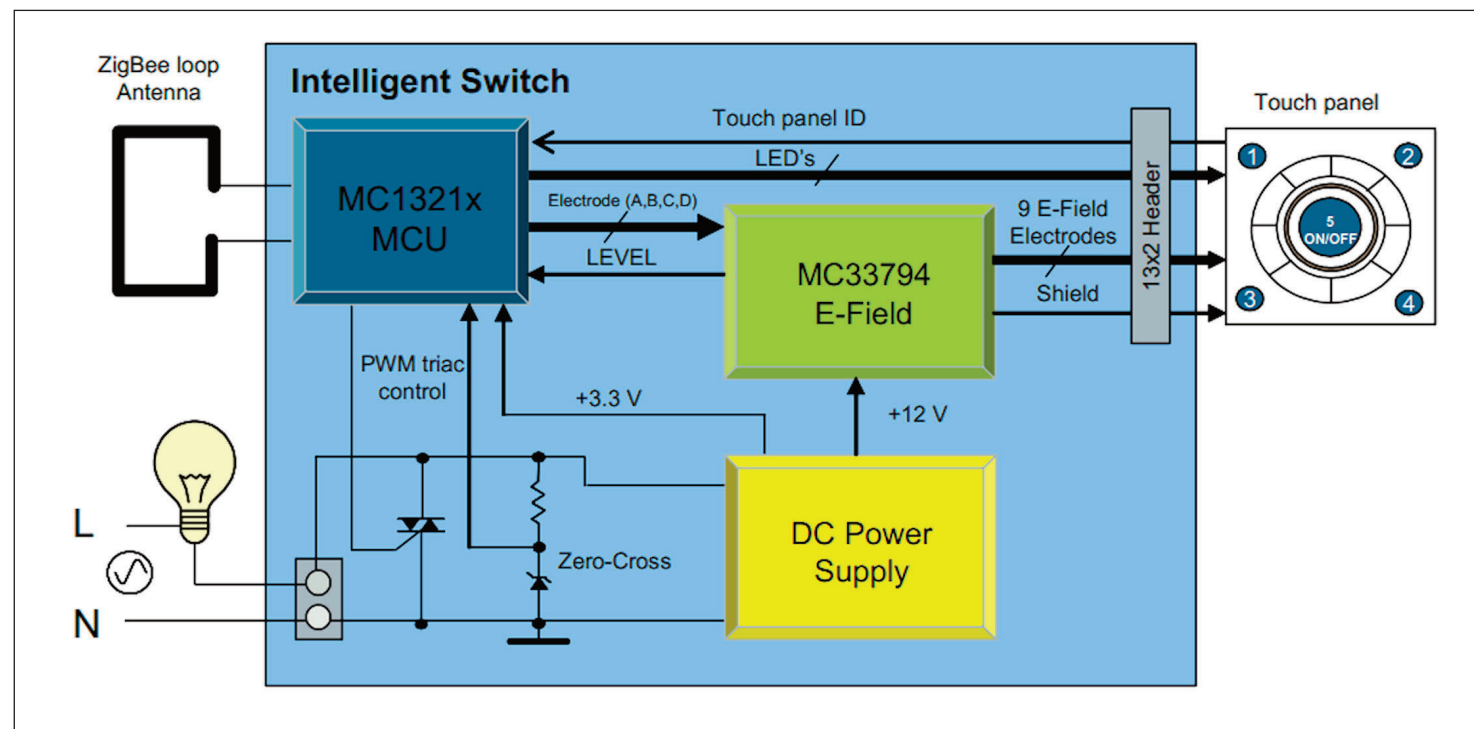



Figura 5: Diagramma a blocchi dello switch intelligente

trollare il flusso delle lampadine in modo tale che vi possa entrare solo la parte d'interesse dell'onda AC con cui è sincronizzato un segnale PWM (Pulse Width Modulation). Oltre al Triac è possibile utilizzare un MOSFET con caratteristiche migliori e più sofisticate rispetto al precedente, infatti esso non chiude il suo canale quando si ha l'attraversamento dello zero (la tensione commuta) ma il tutto dipende soltanto dal controllo della sua tensione di "Gate". Questo è un vantaggio non da poco per realizzare varie opzioni swiching, tuttavia quando viene usato un simile dispositivo per un circuito di dimming, spesso richiede speciali capacità di sincronizzazione con la tensione regolata e quindi circuiti più complicati; in aggiunta il MOSFET è anche spesso più costoso di un Triac, perciò generalmente è preferibile utilizzare il Triac.

TEORIA DI FUNZIONAMENTO: IL CONTROL PANEL

Pannelli di controllo, elettrodomestici, macchinari pesanti, controlli d'illuminazione e tutto ciò che ha pulsanti richiede una sorta di interazione umana per operare; tradizionalmente, i pulsanti sono costituiti da interruttori meccanici che possono deteriorarsi e diventare meno affidabile nel tempo. Gli integrati della Freescale MC34940/MC33794 per la rilevazione del campo elettrico offrono un'alternativa ai pulsanti meccanici. I Circuiti integrati (IC) MC34940/MC33794 contengono la circuiteria necessaria

Figura 6: Vista TOP e BOTTOM board touch panel

Figura 7: Vista TOP della board SI

per generare un campo elettrico a bassa intensità e misurare il suo perturbamento causato da oggetti in movimento dentro e/o fuori da esso. Questo è l'ideale per le applicazioni che desiderano un rilevamento senza contatto, una rilevazione di prossimità e "E-field imaging" tridimensionale. Questi IC hanno il supporto per un microcontrollore e 7-9 elettrodi, che possono essere utilizzati indipendentemente per determinare la dimensione o la posizione di un oggetto in un campo elettrico debole. Poiché hanno la capacità di rilevare il tocco o la prossimità attraverso una superficie isolata e senza un contatto elettrico diretto con elettrodi metallici, il problema di usura, contaminazione e corrosione viene totalmente eliminato. L'integrato MC33794 è destinato all'uso nel rilevamento di oggetti, utilizzando un campo elettrico. Il suo principio di funzionamento è riportato in Figura 4, nella quale si nota come tale IC genera un'onda sinusoidale a bassa frequenza radio, che è impostata da una resistenza esterna ed è ottimizzato per 120 kHz. L'onda sinusoidale ha un basso contenuto di armoniche per ridurre le potenziali interferenze a frequenze più alte e armonicamente connesse. Il generatore interno produce 5V di uscita di picco-picco nominale che viene fatta passare attraverso una resistenza interna di circa 22k . Un multiplexer interno instrada il segnale a uno dei vari morsetti (terminali d'ingresso), simultaneamente un multiplexer in ricezione, collegato con l'elettrodo invia il suo segnale verso un detector che converte un segnale sinusoidale in un segnale continuo (DC). Questo livello DC è filtrato da un condensatore esterno e viene modificato/condizionato per aumentarne la sensibilità, mentre tutte le altre uscite degli elettrodi sono poste a massa dal dispositivo. L'ampiezza e la fase dell'onda sinusoidale dell'elettrodo sono modificate da oggetti in prossimità degli elettrodi. Un condensatore (come una piastra che contiene carica elettrica) si forma tra l'elettrodo e l'oggetto. La tensione misurata è una funzione inversa della capacità tra l'elettrodo da misurare, gli elettrodi circostanti e altri oggetti nel campo elettrico che circonda elettrodo (aumentare il valore capacitivo significa diminuire la tensione). Mediante tale tecnica viene rilevata e interpretata la presenza di un oggetto (o del corpo umano) che perturba lo stato del sistema.

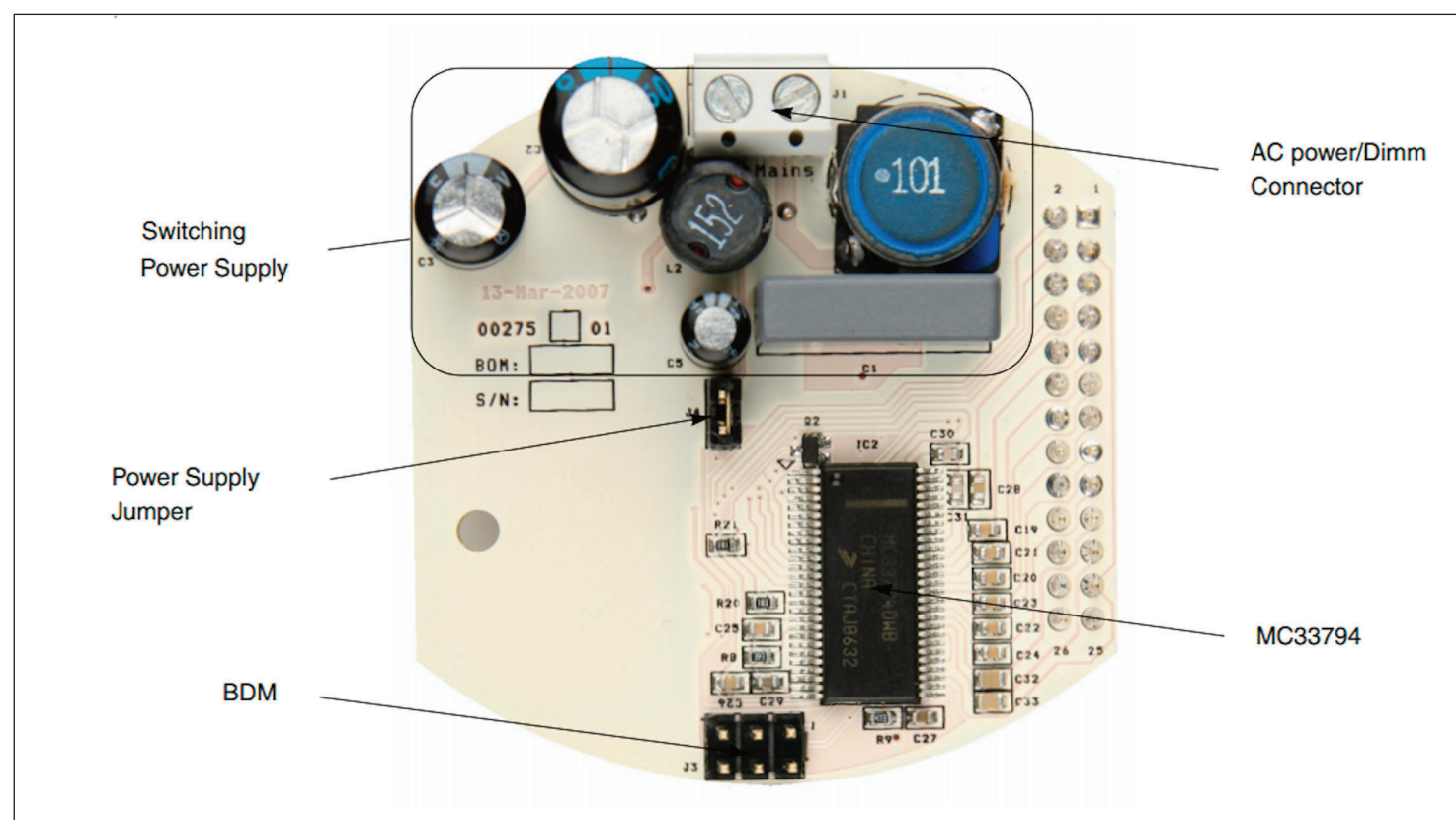



Figura 8: Vista BOTTOM della board SI

Figura 9: Diagramma a blocchi della presa intelligente

SWITCH INTELLIGENTI

Dalla Figura 5 si può subito comprendere come gli interruttori intelligenti possono essere divisi in due blocchi:
Centralina elettronica;
Touch panel.
Il secondo blocco non è altro che la board relativa agli elettrodi, visibile in una sua implementazione in Figura 6, in cui si può notare anche il connettore di collegamento verso la centralina.
Mentre per quanto riguarda il primo blocco, questo è il cuore di tutto il sistema e da Figura 5 si possono vedere le maggiori funzionalità interne e le loro interconnessioni. Come ben si comprende, tutto ruota intorno all'integrato MC13213, contenente un micro processore a 8-bit della famiglia S08 e un modulo wireless Zigbee 802.15.4, tutto in un singolo package. Immediatamente sotto tale blocco distinguiamo quello per la gestione delle lampade mediante il Triac e accanto a quest'ultimo il blocco delle alimentazioni che genera una tensione di 3.3V per l'alimentazione dell'integrato MC13213 e i 12V per MC33794 che si occupa della gestione relativa il touch panel. Infine il sistema prevede un'antenna su PCB per il modulo RF. La Figura 7 riporta il '*Top*' di tale scheda in cui sono visibili i componenti e gli integrati che realizzano le funzioni fino a ora analizzate. Risaltano subito il blocco dell'alimentazione switching (in alto), il microprocessore MC13213 e l'antenna su PCB per le comunicazioni wireless. In Figura 8 invece, è riportato il '*Bottom*' in cui è presente il connettore di riprogrammazione in-circuit (BDM), l'integrato MC 33794 per la gestione del touch panel e l'AC power / dimming connector.

PRESE INTELLIGENTI

Il principio di funzionamento è riportato in Figura 9, da cui si evince che la presa intelligente è molto semplificata

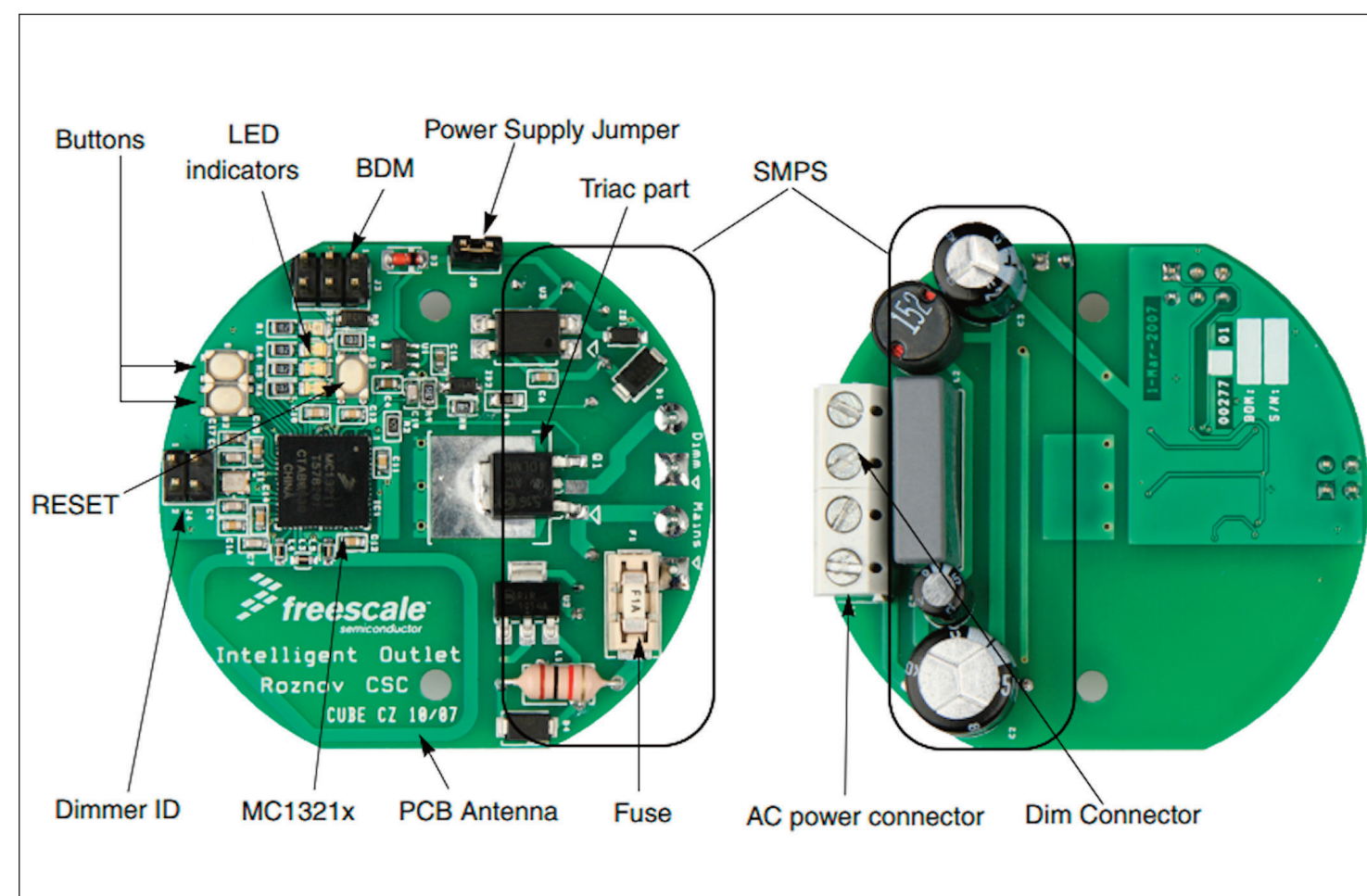



Figura 10: TOP e BOTTOM versione 1 della PI

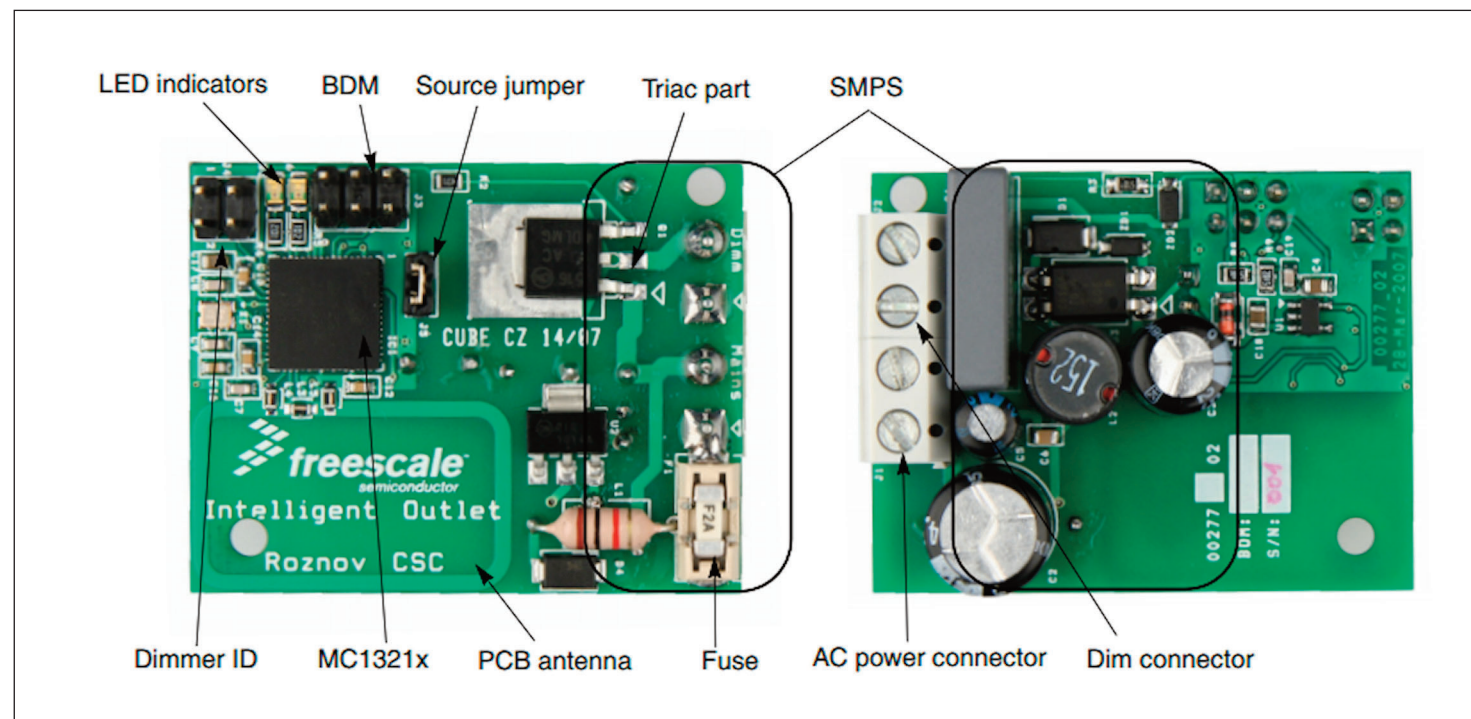



Figura 11: TOP e BOTTOM versione 2 della PI

rispetto allo switch intelligente che invece controlla tutto il sistema e gestisce l'interfaccia con il mondo esterno mediante il touch panel. Di prese intelligenti ne esistono di due tipi, riportate in Figura 10 e Figura 11.
Le differenze tra queste due tipologie sono minime e l'esigenza di questa duplice implementazione è stata dettata della possibilità di fornire due "*case*" differenti (visibili in Figura 1) da poter utilizzare in ambiti diversi com'è stato analizzato all'inizio di quest'articolo. In entrambe le board sono presenti i blocchi funzionali della Figura 9, infatti si nota l'integrato MC13213 per le comunicazioni e la gestione della scheda collegato all'antenna, i componenti per l'alimentazione e per il dimming ma anche il connettore di programmazione. Una cosa interessante da notare è la presenza del connettore AC per le alimentazioni.

CONCLUSIONI

Come visto fino a ora, i vantaggi di tale architettura e di questa implementazione sono molteplici, mediante il sistema (hardware/software) appena analizzato di Freescale, si possono gestire facilmente illuminazioni domestiche con un occhio di riguardo verso i consumi. In aggiunta a tutto questo c'è la comodità di avere un sistema wireless centralizzato e gestibile comodamente con un'interfaccia touch.